AEDC-TR-85-64

Chimera: A Grid-Embedding Technique

J. A. Benek
Calspan Corporation
and
J. L. Steger, F. C. Dougherty, and P. G. Buning
NASA Ames Research Center

April 1986

Final Report for Period November 1, 1980 – October 1, 1985

ARNOLD ENGINEERING DEVELOPMENT CENTER
ARNOLD AIR FORCE STATION, TENNESSEE
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE

APPROVAL STATEMENT

This report has been reviewed and approved.

Keith L. Kushman

KEITH L. KUSHMAN
Directorate of Technology
Deputy for Operations

Approved for publication:

FOR THE COMMANDER

[illegible]

LOWELL C. KEEL, Lt Colonel, USAF
Director of Technology
Deputy for Operations

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AEDC-TR-85-64	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Arnold Engineering Development Center 6b. OFFICE SYMBOL (If applicable) DOT	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Systems Command Arnold Air Force Station, TN 37389-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Arnold Engineering Development Center 8b. OFFICE SYMBOL (If applicable) DO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Air Force Systems Command Arnold Air Force Station, TN 37389-5000	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
65807F			

11. TITLE (Include Security Classification)
SEE REVERSE OF THIS PAGE

12. PERSONAL AUTHOR(S)
Benek, J. A., Calspan Corporation and Steger, J. L., Dougherty, F. C., and (Cont)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 11/1/80 TO 10/1/85	April 1985	129

16. SUPPLEMENTARY NOTATION
Available in Defense Technical Information Center (DTIC).

17. COSATI CODES

FIELD	GROUP	SUB. GR.
20	04	
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
grid-embedding techniques
computational fluid dynamics
computer programs (Cont)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A three-dimensional chimera grid-embedding technique is described. The technique simplifies the construction of computational grids about complex geometries. The method subdivides the physical domain into regions which can accommodate easily generated grids. Communication among the grids is accomplished by interpolation of the dependent variables at grid boundaries. The procedures for constructing the composite mesh and the associated data structures are described. The method is demonstrated by solution of the Euler equations for transonic flow about three ellipsoid bodies in close proximity, a wing/body, and a wing/body/tail.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ [✓ SAME AS RPT.] DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL W. O. Cole	22b. TELEPHONE NUMBER (Include Area Code) (615)454-7813 22c. OFFICE SYMBOL DOS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

11. TITLE

Chimera: A Grid-Embedding Technique

12. PERSONAL AUTHORS (Concluded)

Buning, P. G., NASA Ames Research Center

18. SUBJECT TERMS (Concluded)

chimera-grid solution
finite-difference techniques
transonic flow
domain-decomposition techniques
grid-adapting methods

PREFACE

The work reported herein was conducted by the Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC). The Program Manager was Dr. Keith Kushman. The results were obtained by Calspan Corporation, AEDC Division, operating contractor for the aerospace flight dynamics testing facilities at the AEDC, AFSC, Arnold Air Force Station, Tennessee 37389-5000.The research was conducted in the Propulsion Wind Tunnel Facility (PWT) during the period November 1, 1980 through October 1, 1985 under AEDC Project Numbers DB205PW and DB25PW, and the manuscript was submitted for publication October 1, 1985.

This work was the result of a joint effort with NASA Ames Research Center personnel, who provided computer support and technical guidance regarding the implementation of the embedded mesh concept. The authors also acknowledge the contribution of Dr. J. C. Erickson, Jr. for his technical criticism of the work described in this report.

CONTENTS

ILLUSTRATIONS

Figure Page

1.0 INTRODUCTION

Solution of the partial differential equations of fluid motion by finite-difference techniques requires that the computational domain and dependent variables be represented on a network of discrete points. The distribution of these points is influenced by the choice of the coordinate system, the order of the numerical approximation, and the location of strong geometric and flow-field gradients. Typically, body-fitted, curvilinear coordinate systems are used to simplify the application of boundary conditions. Construction of grids with the requisite smoothness and point clustering remains one of the most nettlesome tasks associated with the solution of the equations of fluid motion. This is especially true for three-dimensional (3-D) configurations as the effort required to generate an acceptable mesh increases rapidly with increasing geometric complexity and quickly becomes prohibitive. The considerable effort (e.g., Refs. 1 through 4) that has been devoted to the development of reliable methods to mitigate these difficulties may be broadly put into two groups, (1) domain-decomposition and (2) grid-adapting methods.

Domain-decomposition techniques subdivide the computational domain into simpler subdomains which admit a more easily constructed mesh. Several strategies have been explored to subdivide the domain and establish communications among the subdomains. One group of approaches, the grid-patching or zonal methods, uses common or shared boundaries and another uses embedded or overset grids to subdivide the domain. The work of Rubbert and Lee (Ref. 5) and Lee (Ref. 6) is typical of the methods which construct a global mesh from subdomains which share common boundaries. They generate a global mesh by solving grid-generation equations on all subdomains simultaneously and by requiring that the grid lines be continuous across subdomain boundaries. A difficulty with this approach is that irregularities which occur in corners and along boundaries impose constraints on the algorithm used to solve the flow equations. Lasinski et al. (Ref. 7) take an alternate approach and solve for the flow field on each subdomain separately with communication among the grids established by the transfer of boundary data. In their approach, the patches overlap one point with common points on the boundary to obviate interpolation for boundary data. Hessenius and Pulliam (Ref. 8) have modified the approach of Ref. 7 to allow characteristic boundary conditions to be applied at subdomain boundaries. Rai (Ref. 9) further generalized the method to admit independent grids in each subdomain. Communication across grid boundaries is accomplished by means of special difference formulae at the boundaries which maintain conservation properties across the subdomains. Similar methods have been developed by Miki and Takagi (Ref. 10). Holst et al. (Ref. 11) have applied the technique to large 3-D grids.

The grid-embedding or oversetting techniques do not require common boundaries between subdomains, but rather, a common or overlap region is required to provide the means of matching the solutions across boundary interfaces. The usual procedure uses interpolation of embedded boundaries to provide the necessary communication among the grids. Atta (Ref. 12) and Atta and Vadyak (Ref. 13) employed this approach to solve the full-potential equation in two and three dimensions. Their implementations used a separate implicit solution algorithm for each mesh. Steger et al. (Ref. 14), Benek et al. (Ref. 15), and Benek et al. (Ref. 16) developed a "chimera" scheme in two and three dimensions for the solution of both a linearized flow model and the Euler equations. Lombard and Venkatapathy (Refs. 17 and 18) use overset grids aligned with shock waves to produce highly resolved solutions of inlet flows. Fuchs (Ref. 19) applied the method to internal flows, and Rai (Ref. 20) uses a combination of patched and overset grids to solve rotor-stator interactions. Dougherty (Ref. 21) and Dougherty et al. (Ref. 22) have extended the grid-embedding technique to allow movement of embedded grids to follow time-dependent motions. A closely related approach developed by Wedan and South (Ref. 23) employs a global Cartesian mesh in which the body is embedded. Grid points that lie within the body are located and automatically excluded from the solution process.

The second technique, grid-adapting methods, causes the mesh to evolve with the solution of the flow equations. These methods seek to make the most efficient use of available mesh points, as well as to reduce the grid-generation effort by automatically clustering grid points to regions of high gradient. An advantage of the method is that the initial mesh does not need to anticipate accurately all regions of large flow gradients. There are several implementations of the method. Gnoffo (Ref. 24) models the mesh as a network of springs whose constants are determined from the flow gradients. Nakahashi and Deiwert (Ref. 25) extend this idea to allow both linear and torsional springs and have applied the method to both steady and unsteady flow problems. Ghia et al. (Ref. 26) couple the grid-evolution equation to the flow equation by requiring that the coefficient of the convective term in the flow model be minimized. Brackbill (Ref. 27) and Saltzman and Brackbill (Ref. 28) use variational techniques to produce grid-evolution equations. Berger (Ref. 29) and Berger and Oliger (Ref. 30) developed a dynamic grid refinement technique which embeds successively finer grids to resolve flow gradients as they develop in the solution process. Unfortunately, the adaptive techniques have not been sufficiently developed to allow an assessment of their applicability to general 3-D flows.

This report documents the development of the chimera grid-embedding technique described in Refs. 14, 15, and 16. We chose the grid-embedding approach for solution of complex 3-D flows because it provides the flexibility to employ boundary-conforming grids on component parts of the geometry, to refine the mesh selectively in regions of interest, and to permit the solution of different flow models on the component grids. Because of its structural diversity, we call our implementation a chimera scheme after the creature from Greek mythology which is compounded of incongruous parts. The method is a generalization of the versatile grid-patching/zonal approach, and therefore, includes their advantages. Thus, advances made in the latter approach have an immediate counterpart in the chimera technique. Although the chimera approach allows different flow models to be solved on each subdomain (e.g., Refs. 11 and 31), the present implementation is restricted to the solution of the Euler equations on each grid.

2.0 GENERAL DESCRIPTION

Domain-decomposition techniques have two principal elements, (1) decomposition of the computational domain into subdomains and (2) communication among the subdomains. In the chimera approach, each subdomain requires a separate, independent grid generation by any acceptable technique. Each subdomain is chosen to lessen the effort required to construct an acceptable mesh, and perhaps, to isolate a particular region of the flow (e.g., where viscous effects are important). As explained in Section 3.2, the chimera implementation increases the flexibility of subdomain selection by removing regions of a mesh common to an embedded grid. That is, an embedded mesh introduces an artificial boundary or ''hole'' into the mesh in which it is embedded. Because the regions interior to the hole do not enter into the solution process, intergrid communication is simplified since communication among the grids is restricted to the transfer of boundary data. Appropriate boundary values are interpolated from the mesh or meshes in which the boundary is embedded. The chimera procedure naturally separates into two parts, (1) generation of the composite mesh and associated interpolation data and (2) solution of the flow model or models on the composite mesh. Each part is embodied in a separate computer code, PEGSUS and XMER3D. PEGSUS takes the independently generated component grids and the embedding structure as input and automatically constructs the composite mesh and interpolation data which are output. XMER3D takes the PEGSUS output and flow specifications as input and solves the appropriate flow equations on each grid.

3.0 PEGSUS

Automatic generation of a composite mesh from the input component grids requires PEGSUS to (1) establish the proper lines of communication among the grids through appropriate data structures, (2) construct holes within grids, (3) identify points within holes, (4) locate points from which boundary values can be interpolated, and (5) evaluate interpolation parameters. In addition, PEGSUS performs consistency checks on interpolation data to assure its acceptability and constructs output files with the data structure used in XMER3D. A structure chart of PEGSUS is given in Appendix A.

3.1 EMBEDDING HIERARCHY

The data structures required to manage the flow of data among the grids can become cumbersome unless some restriction is placed upon the allowable interactions. A hierarchical form follows naturally from the embedding process; embedded grids occupy a lower level of the hierarchy than the grids in which they are embedded. Hierarchical forms also have a convenient mathematical representation as graphs. Such a representation greatly simplifies the development of data structures required to manipulate the transfer of data among the grids by identifying the communication links that must be established. To facilitate the discussion, introduce the following nomenclature: designate grids which comprise level ℓ of the hierarchy as $G_{\ell,i}$, $i = 2, \ldots$ In general, grids on a given level ℓ are embedded within grids on level $\ell - 1$, overlap other grids on level ℓ, and have one or more grids on levels $\ell + 1$ embedded within them. Such an arrangement is shown in Fig. 1. The figure includes the corresponding graph (See Section 3.4). The lines connecting the grids indicate the intergrid communication links that must be supported by the data structures and suggest the complexity involved.

It is not necessary to include all the interactions shown in Fig. 1. Very general configurations can be considered with a restricted hierarchy at the expense of additional labor in the construction of the component grids. The interactions permitted by the hierarchy adopted for the work described in this report are shown in Fig. 2; grids on level ℓ are constrained to be completely contained within a single grid on level $\ell - 1$, and grids on level ℓ must be disjoint. The major advantages of the adopted structure are the simplifications it provides in the construction of the data structures and in the limitations on the searches required to locate points in other grids which may serve for interpolation of boundary data.

3.2 HOLE GENERATION

Because each component mesh is generated independently, complications frequently arise when the grids are embedded. For example, points of an enclosing mesh, $G_{\ell,i}$, may be found to lie within a solid boundary contained within an embedded grid, $G_{\ell+1,j}$. Such points lie out of the computational domain and must be excluded from the solution process. In addition, a large number of points must be interpolated if every point common to $G_{\ell,i}$ and $G_{\ell+1,j}$ (i.e., $G_{\ell+1,j} \cap G_{\ell,i}$) is to be updated. Lombard and Venkatapathy (Ref. 17) found that such extensive interpolation can degrade global accuracy when there is a considerable difference in mesh cell size (i.e. spatial resolution) between the grids $G_{\ell,i}$ and $G_{\ell+1,j}$. To avoid such complications, only the boundary of each embedded grid is updated; the points of $G_{\ell,i}$ contained within a subregion of $G_{\ell+1,j}$ are excluded from the solution on $G_{\ell,i}$. Thus, the embedded mesh, $G_{\ell+1,j}$ introduces an artificial boundary or "hole" into $G_{\ell,i}$. The only computational requirement is that there remains a sufficient overlap (i.e., points in $G_{\ell,i} \cap G_{\ell+1,j}$ and exterior to the hole) to support an interpolation for the outer boundary of $G_{\ell+1,j}$ from points in $G_{\ell,i}$ (Fig. 3). Similarly, the overlap must be sufficient to allow the hole boundary in $G_{\ell,i}$ to be interpolated from points in $G_{\ell+1,j}$. A minimum overlap exists that is dependent on the type of interpolation used.

A hole is constructed as follows: a surface, C, is introduced into $G_{\ell,i}$. In general, the surface encloses solid boundaries contained within the embedded grid $G_{\ell+1,j}$ and serves as an initial hole boundary. Whenever boundary-conforming component grids are employed, the simplest choice for C is a level surface of $G_{\ell+1,j}$. A search of $G_{\ell,i}$ locates points interior to C. These points are "marked" for future reference by changing the value of an integer array, IBLANK, corresponding to these points, from 1 to 0.

Figure 4 illustrates the details of the search procedure in two dimensions. The procedure is as follows: (1) Define the initial hole boundary by a level curve in $G_{\ell+1,j}$ (Fig. 4a). (2) Construct outward normals, $\vec{N}$, at each point, P_c, defining C (Fig. 4b). (3) Determine a temporary origin, say P_o, located within C by averaging the coordinates. (4) Define a "search" circle about P_o with radius R_{max}, where R_{max} is the maximum distance from P_o to points on C (Fig. 4c). (5) Test the magnitude of $\vec{r}$, the position vector relative to P_o, for every point P of $G_{\ell,i}$. If $|\vec{r}| \geq R_{max}$, P lies outside the search circle and hence need not be considered further. Whenever $|\vec{r}| < R_{max}$, P falls within the search circle and additional testing is required. (6) Compute $\vec{N} \cdot \vec{R}_p$ where $\vec{N}$ is the outward normal at the point P_c on C closest to P, and $\vec{R}_p$ is the position vector to P from P_c (Fig. 4d). If $\vec{N} \cdot \vec{R}_p \geq 0$, P is outside C; if $\vec{N} \cdot \vec{R}_p < 0$, P is inside C and IBLANK corresponding to this point is set to 0.

The points of $G_{\ell,i}$ within the hole are excluded from the solution and are not usable as boundary points. Therefore, additional points of $G_{\ell,i}$ are identified as hole-boundary or fringe points. Values of the unknowns at these boundary points will be interpolated from the embedded mesh, $G_{\ell+1,j}$. The boundary points are constructed from points in $G_{\ell,i}$ which are not hole points but which have nearest neighbors that are. Figure 5 illustrates the boundary construction. The procedure is to examine the nearest neighbors (Fig. 5) of each point, P, in $G_{\ell,i}$ at which IBLANK = 1. If a neighbor is a hole point, P is a boundary point. The indices of the fringe points are added to a list of boundary points which will require interpolated data. The boundary point is also temporarily "marked" by setting the value of IBLANK to a nonpositive number which PEGSUS associates with $G_{\ell+1,j}$. Once all the hole boundaries in $G_{\ell,i}$ are constructed, the fringe point indices are added to a list of boundary points which will require interpolated data. (Refer to Appendix B for details of the associated data structures.) To simplify the logic of XMER3D, the values of IBLANK corresponding to the boundary points are reset to 0.

The primary function of the hole or artificial boundary introduced into a mesh, $G_{\ell,i}$, by an embedded mesh, $G_{\ell+1,j}$, is to exclude a region of $G_{\ell,i}$ which may fall within solid boundaries contained in $G_{\ell+1,j}$. It is possible for the reverse condition to exist. A region of the mesh $G_{\ell+1,j}$ may fall within solid boundaries of $G_{\ell,i}$. A two-dimensional (2-D) example is given in Fig. 6. Because the mesh about Body 2 overlaps Body 1 in mesh $G_{\ell,i}$ in which it is embedded, a region of $G_{\ell+1,j}$ about Body 1 must be excluded from the solution on $G_{\ell+1,j}$. The procedure for constructing such a hole boundary is similar to that described above. The examples of the chimera scheme given in this report avoid such complications. The interested reader is directed to Refs. 21 and 22 for examples which include holes induced in embedded grids.

3.3 INTERPOLATION

As the separate grids are to be treated as independent entities, boundary conditions must be supplied to each. The boundary conditions of the differential equations which model the flow provide data only at the boundaries of the computational domain. Thus, other data must be obtained for the subdomain boundaries which are not coincident with those of the computational domain. Because the subdomain boundaries typically lie in the interior of the computational domain where the differential equations are valid, it seems appropriate that the solution of these equations should provide the necessary boundary data. There are currently several approaches (e.g., Refs. 15, 16, 20, 29, and 31) to obtain these data, but all involve some form of interpolation of data in one mesh to provide the necessary data to another.

Experience with a 2-D application of the chimera grid-embedding scheme (Ref. 15) indicates that difficulties can arise when a shock crosses grid boundaries. Figure 7 makes a comparison of pressure distributions obtained from solution of the Euler equations about a supercritical airfoil on a single mesh and on a chimera grid. There is a mismatch in the solutions in the neighborhood of the expansion preceding the shock. Examination of the Mach number contours of the chimera-grid solutions (Fig. 8) reveals a considerable amount of mismatch and hash in the overlap region near the shock/grid-boundary intersection. Several factors could contribute—the nonconservative nature of the Taylor series interpolation; the reflecting boundary conditions imposed at the overlap; and the meager extent of the overlap in the vicinity of the shock (Fig. 9).

There is much disagreement about the proper interpolation method to employ, especially for cases in which shock waves or other regions of high gradients cross grid boundaries. The basis of concern is that interpolation schemes assume continuity of the interpolant. Regions of high gradient require approximations to the higher derivatives of the solution, terms which are typically neglected. As a result, several methods have been developed which modify the numerical difference procedure applied at grid interfaces to maintain a proper representation of fluxes at the boundaries. The schemes proposed by Hessenius and Pulliam (Ref. 8) and Rai (Ref. 9) are typical of such methods. Berger (Ref. 32) developed a generalized difference scheme based upon the concept of a weak solution to the differential equations and has the advantage that continuity of the interpolant is not required. The method was illustrated for the case of 2-D overlapping grids.

Several investigations have found factors other than the nonconservative interpolation to be important. Eberhardt (Ref. 33) examined shock/grid-boundary interactions between embedded grids. He found that a major factor in the interaction was the boundary conditions imposed at the overlap boundaries. Characteristic or nonreflecting boundary conditions reduced the mismatch at the shock, improved accuracy, and increased the convergence rate for both first- and second-order Taylor series interpolation. (Similar results are reported in Ref. 8.) Lombard and Venkatapathy (Ref. 17) found that a disparity in spatial resolution can significantly affect the shock/grid-boundary interaction. They also found the proximity of the boundaries (i.e. overlap) to be important, particularly whenever a significant difference in mesh resolution exists.

Mastin and McConnaughey (Ref. 34) studied computational problems associated with interpolation on composite grids. They showed that bilinear interpolation in two dimensions is superior to a Taylor series expansion when higher order derivatives of the solution are not important. They also found that a two-cell overlap was sufficient to provide accurate interpolation whenever the cell sizes of the overlapped grids are comparable. Simple computations (Ref. 35) of the flow variables interpolated across an oblique shock on a rectangular grid indicate that bilinear interpolation is superior to Taylor's series interpolation.

We employ a trilinear interpolation for the 3-D chimera scheme. This method provides an additional advantage of a more compact stencil. We also attempt to maintain a four- to five-cell overlap between grid boundaries. The interpolation scheme is coupled with nonreflective boundary conditions on the grid boundaries. (For more details of the interpolation, see Appendix C.) However, it remains possible that nonconservative effects may be important for the acuracy of solutions near shock/grid-boundary interactions, particularly in cases for which the spatial resolution between grids is comparable and in cases with strong shocks.

3.4 DATA STRUCTURES

A chimera method requires the management of a large amount of grid and solution information. It is necessary to keep track of the storage locations of the coordinates of each grid, solution data on each grid, interpolation points, interpolated data, interpolation stencils, points within holes, and the relationships among the grids in the hierarchy. In addition, the management function must be implemented in an automatic manner that is transparent to the user. Fortunately, the computer scientists have developed the required management techniques (e.g., Refs. 36 and 37). Unfortunately, implementing these techniques in FORTRAN is not always straightforward.

Concepts which can be helpful for use with the data structure draw on ideas from the theory of graphs (e.g., Ref. 38). The graphical representation of the grid-embedding hierarchy used in this report is shown in Fig. 2. Each grid $G_{\ell,i}$ is called a node of the graph. However, a node may contain much more information than just the grid coordinates. Additional information we have associated with each node includes grid-storage location; number of points in each coordinate direction; location of the grid in the embedding hierarchy; the precursor grid, $G_{\ell-1,k}$; number of descendants, $G_{\ell+1,j}$, $j = 1, 2, \ldots$; location of descendants; location of interpolation stencils; location of interpolation coefficients; location of interpolated boundary data; and location of hole or excluded points. Most of these data can be stored in lists; connections among the data are made through the use of linked lists, and pointers (See Refs. 36 and 37). To illustrate, consider the management of a single array containing spatial coordinates. The data are stored in a stacked or sequential manner. Thus, it is only necessary to store the position of the first element (a pointer) in each grid and the number of points in each coordinate direction on each grid (a linked list) to locate the coordinates of any point. Details of the data structures are contained in Appendix D.

Other approaches to the data structures are possible. For example, Norton et al. (Ref. 39) use a generalized grid structure similar to that used in finite-element methods. Each computational cell (or grid point) is numbered in an arbitrary manner as opposed to the usual systematic ordering of points employed by finite-difference techniques. Instead, lists of the indices of the points used in the finite-difference representation of each of the flow equations are maintained. This procedure allows very different topologies to be used on component grids and implicit interpolation for grid interfaces to be incorporated into the scheme. The major disadvantage of the method is the large storage overhead required for the lists of points in the difference stencil.

4.0 XMER3D

The implementation of the chimera scheme must provide for the use of multiple flow models. The simplest choice of models is the Euler equations for inviscid flow and the Navier-Stokes equations for viscous flow. For purposes of demonstration of the method, we solve only the Euler equations. However, because of the choice of numerical algorithm, the extension to viscous flow is straightforward. The 3-D Euler equations are solved by the implicit, approximate factorization algorithm of Beam and Warming (Refs. 40 and 41). The implementation follows that of Steger (Ref. 42) and Pulliam and Steger (Ref. 43). These formulations use explicit boundary conditions which provide a convenient method of imposing the correct boundary conditions on the various grids with minimal changes to the code. A version of the code (Benek, J. A. ''Vectorized Implicit Algorithm for Solution of the Navier-Stokes Equations,'' unpublished, 1979) vectorized for the Cray computer served as the basis for XMER3D (Appendixes D, E, and F).

4.1 THE SOLUTION ALGORITHM AND GRID HOLES

Grid points that belong to a hole must be excluded from the solution. Once the points have been located and identified (i.e. IBLANK = 0), it is a simple matter to modify the implicit algorithm. The modification required is illustrated by considering the algebraic system typical of numerical approximation of the flow equations.

$$A\phi = F \tag{1}$$

where A is the coefficient matrix assumed to be tridiagonal for convenience, ϕ is the vector of unknowns, and F is a known vector. Let ϕ_3, ϕ_4, and ϕ_5 be elements belonging to a hole, and let f_3, f_4, and f_5 be values specified for ϕ_3, ϕ_4, and ϕ_5. The equations may be partitioned to isolate the hole

$$\left[\begin{array}{cc:ccc:cc} a_{11} & a_{12} & 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ a_{21} & a_{22} & a_{23} & 0 & 0 & & \\ \hdashline 0 & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & 0 \\ \hdashline \multicolumn{2}{c}{\multirow{2}{*}{0}} & 0 & 0 & a_{65} & a_{66} & a_{67} \\ & & 0 & 0 & 0 & a_{76} & a_{77} \end{array}\right] \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \cdots \\ \phi_6 \\ \phi_7 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ \cdots \\ F_3 \\ F_4 \\ F_5 \\ \cdots \\ F_6 \\ F_7 \end{Bmatrix} \qquad (2)$$

The desired modification must allow the application of the standard tridiagonal solution algorithm and must not interfere with vectorization. These criteria may be satisfied if a_{32}, a_{34}, a_{43}, a_{45}, a_{54}, and a_{56} (i.e. the off-diagonal elements) are set to zero; a_{33}, a_{44}, and a_{55} (the diagonal elements) are set to the identity; and F_3, F_4, and F_5 are set to f_3, f_4, and f_5. The system takes the form

$$\left[\begin{array}{cc:ccc:cc} a_{11} & a_{12} & 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ a_{21} & a_{22} & a_{23} & 0 & 0 & & \\ \hdashline & & 1 & 0 & 0 & & \\ \multicolumn{2}{c:}{0} & 0 & 1 & 0 & \multicolumn{2}{c}{0} \\ & & 0 & 0 & 1 & & \\ \hdashline \multicolumn{2}{c}{\multirow{2}{*}{0}} & 0 & 0 & a_{65} & a_{66} & a_{67} \\ & & 0 & 0 & 0 & a_{76} & a_{77} \end{array}\right] \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \cdots \\ \phi_6 \\ \phi_7 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ \cdots \\ f_3 \\ f_4 \\ f_5 \\ \cdots \\ F_6 \\ F_7 \end{Bmatrix} \qquad (3)$$

The hole logic can be incorporated into the algorithm so that vectorization is maintained. The array IBLANK is defined for each point of the grid. If a point is within a hole, IBLANK = 0; otherwise IBLANK = 1. A simple set of switches can be constructed that automatically multiply each row of the coefficient matrix, A, by IBLANK

$$a_{ij} = a_{ij} \bullet \text{IBLANK}_i \qquad i \neq j$$

$$a_{ij} = a_{ij} \bullet \text{IBLANK}_i + (1 - \text{IBLANK}_i), \qquad i = j$$

The known vector, F, can similarly be modified as

$$F_i = F_i \bullet IBLANK_i + (1 - IBLANK_i) \bullet f_i$$

For applications to the Beam and Warming algorithm, ϕ corresponds to corrections to the latest approximation Φ^n of the solution to the flow equation and

$$\Phi_i^{n+1} = \Phi_i^n + \phi_i^n \tag{4}$$

Thus, the specified values of F_i in the holes are zero (i.e., $f_i = 0$, i = hole point). Since the values of Φ_3^n and Φ_5^n are determined by interpolation from the solution on another mesh, and since the modified algorithm automatically produces ϕ_3, ϕ_4, $\phi_5 = 0$, the interpolated boundary values Φ_3^n and Φ_5^n are automatically preserved.

4.2 CONVERGENCE ACCELERATION

Additional changes to the solution algorithm were made to improve the convergence rate. These include modifications to the numerical dissipation terms and modification of the time step. The explicit dissipation term in Ref. 43 has the form

$$\epsilon J^{-1} [(\nabla\Delta)_\xi^2 + (\nabla\Delta)_\eta^2 + (\nabla\Delta)_\zeta^2] J\vec{Q} \tag{5}$$

where J is the Jacobian of the coordinate transformation; $\vec{Q} = [\varrho, \varrho u, \varrho v, \varrho w, e]/J$ is the vector of dependent variables; density, ϱ; momentum components (ϱu, ϱv, ϱw); total energy, e; and ϵ is a user-supplied constant of $0(\Delta t)$, where Δt is the time step. Equation (5) was initially modified to

$$\epsilon J^{-1} [\psi_\xi(\nabla\Delta)_\xi^2 + \psi_\eta (\nabla\Delta)_\eta^2 + \psi_\zeta (\nabla\Delta)_\zeta^2] J\vec{Q} \tag{6}$$

where

$$\psi_\xi = | \xi_x | + | \xi_y | + | \xi_z | \tag{7a}$$

$$\psi_\eta = | \eta_x | + | \eta_y | + | \eta_z | \tag{7b}$$

$$\psi_\zeta = | \zeta_x | + | \zeta_y | + | \zeta_z | \tag{7c}$$

and ξ, η, and ζ are the curvilinear computational coordinates. The dissipation remains in a nonconservative form but is weighted by an approximation to the eigenvalues of the Jacobians of the flux terms (Refs. 44 and 45). The approximation has the advantage of avoiding the evaluation of square roots. A similiar approximation to the implicit smoothing term of Ref. 43 preserves the block tridiagonal structure of the difference equations. The 3-D solutions presented here were obtained with that form of the smoothing.

However, additional experimentation with the smoothing terms showed that improved convergence rates can be obtained with the combined form

$$\partial_\xi^2 (\nu_\xi \vec{Q}) + \partial_\eta^2(\nu_\eta \vec{Q}) + \partial_\zeta^2 (\nu_\xi \vec{Q}) \tag{8}$$

where, e.g.,

$$\nu_\xi = \begin{cases} \epsilon \left[\dfrac{\lambda_\xi}{J}\right] \partial_\xi^2 (J\vec{Q}) & \text{if } \epsilon \geq \epsilon_s \mu_\xi \\ \epsilon_s \mu_\xi \left[\dfrac{\lambda_\xi}{J}\right] & \text{if } \epsilon < \epsilon_s \mu_\xi \end{cases} \tag{9}$$

$$\lambda_\xi = (1 + \| \vec{V} \|)(| \xi_x | + | \xi_y | + | \xi_z |) \tag{10a}$$

$$\lambda_\xi = (1 + \| \vec{V} \|) (| \eta_x | + | \eta_y | + | \eta_z |) \tag{10b}$$

$$\lambda_\xi = (1 + \| \vec{V} \|) (\zeta_x | + |\zeta_y | + |\zeta_z |) \tag{10c}$$

and where

$$\mu_\xi = \frac{\partial_\xi^2 \varrho}{<\varrho>} = 4 \frac{|\varrho_{j+1} - 2\varrho_j + \varrho_{j-1}|_{k,\ell}}{|\varrho_{j+1} + 2\varrho_j + \varrho_{j-1}|_{k,\ell}} \tag{11a}$$

$$\mu_\eta = \frac{\partial_\eta^2 \varrho}{<\varrho>} = 4 \frac{|\varrho_{k+1} - 2\varrho_k + \varrho_{k-1}|_{j,\ell}}{|\varrho_{k+1} + 2\varrho_k + \varrho_{k-1}|_{j,\ell}} \tag{11b}$$

$$\mu_\zeta = \frac{\partial_\zeta^2 \varrho}{<\varrho>} = 4 \frac{|\varrho_{\ell+1} - 2\varrho_\ell + \varrho_{\ell-1}|_{j,k}}{|\varrho_{\ell+1} + 2\varrho_\ell + \varrho_{\ell-1}|_{j,k}} \tag{11c}$$

The corresponding implicit term has the form

$$\partial_\xi^2 [\epsilon_i(\frac{\lambda_\xi}{J}) + \epsilon_s \mu_\xi (\frac{\lambda_\xi}{J})] \, J \, \vec{q} \tag{12}$$

where $\vec{q}$ is the intermediate correction vector for the ξ direction, and where ϵ, ϵ_s, and ϵ_i are user-supplied constants. Typically, $\epsilon = 0.05$, $\epsilon_s = 1.0$, and $\epsilon_i = 0.15$.

The second modification to Ref. 43 is the addition of local time stepping. At each point the time increment, Δt, was replaced by a local time step, h, in the form

$$h = \Delta t_{lim}/(1 + \sqrt{J}) \qquad (13)$$

where Δt_{lim} is a limiting value at Δt; generally, $\Delta t_{lim} \leq 5$. This form, suggested by Pulliam (Ref. 46), greatly speeds the convergence of single-mesh solutions. We employed it on the assumption that it would also enhance the convergence rate of the embedded grid as well. Holst et al. (Ref. 11) used Eq. (13) for the local time step. Flores (Ref. 47) presented computational results that show excellent convergence rates can be obtained. He also indicated that the diagonal form of the Beam and Warming algorithm implemented by Pulliam and Chaussee (Ref. 48) can achieve improved convergence with a fourth-order implicit smoothing term. A fourth-order, implicit term destroys the tridiagonal nature of the algorithm and produces a pentadiagonal form. This is not a severe penalty for the scalar inversions used in the diagonal algorithm. The tridiagonal form was retained here because the method will also be applied to time dependent problems (e.g., Refs. 21 and 22), and the diagonal form (Ref. 48) is not conservative in time.

An alternate form for the local time step was investigated. The new form uses a more exact approximation of the local Courant number; it is

$$h = \frac{CFL}{\lambda} \qquad (14)$$

where CFL is the local Courant number which is a user-supplied input for each mesh, and

$$\lambda = (\lambda_\xi + \lambda_\eta + \lambda_\zeta)/3 \qquad (15)$$

where λ_ξ, λ_η, and λ_ζ have been defined previously. Equation (13) was found to provide the most consistent acceleration for arbitrary combinations of grids. For the cases tested, it was found that $CFL < 2.0$.

5.0 APPLICATION AND VERIFICATION

The motivation for development of the chimera scheme is the simplification of grid generation for computational problems involving complex geometry. In particular, the requirement for routine computation of the effects of the wind tunnel environment on aerodynamic models at the AEDC (Ref. 49) has emphasized the importance of a simple method for 3-D grid generation. In addition, there is a requirement for computations of time-dependent problems involving aerodynamic configurations in relative motion as exemplified in the space shuttle booster separation and store separation from fighter aircraft. Because of the complexity inherent in the grid-embedding process, particularly with the associated data structures, we decided to first test the general concepts in two dimensions. The lessons learned from this initial development step proved to be invaluable to the extension of the procedure to three dimensions. Therefore, the following first summarizes the 2-D results and notes the lessons which were found to be significant for the development of the 3-D procedure. Then, the 3-D results are presented.

5.1 2-D APPLICATIONS

The chimera scheme was initially demonstrated in two dimensions using a linear, incompressible flow model (Ref. 14). The method was extended to solution of the Euler equations about three, more complex geometries (Ref. 15). The first was a circular cylinder which served as a check of the method since an analytic solution exists for the incompressible case. The second was a supercritical airfoil with a shock wave crossing grid boundaries. This example was used to explore possible difficulties with a shock/grid-boundary interaction. The third was a flapped supercritical airfoil which served to illustrate the method with a complex geometry.

5.1.1 Circular Cylinder

The flow about a circular cylinder in crossflow was computed for two Mach numbers. A two-level grid hierarchy was used (Fig. 10); the first level consisted of a 51 by 51 stretched Cartesian grid; and the second level was an 85 by 30 polar grid that contained the cylinder. The first calculation of cylinder surface pressure for $M_\infty = 0.25$ was found to agree very well with a potential solution with a Prandtl-Glauert compressibility correction and with an Euler solution on a 90 by 70 polar mesh as shown in Ref. 15. A second, supercritical, calculation was made for $M_\infty = 0.50$. The vortex phenomena described by Salas (Ref. 50) were observed as expected. The resulting solution was found to produce an asymmetric shedding of vortices. The shed vortices passed freely downstream through the polar grid boundary. Only the dispersion caused by the change in mesh resolution was noted.

5.1.2 Airfoil

The second geometry was the Dornier SKF1.1 supercritical airfoil (Ref. 51) with a modified, sharp trailing edge geometry. Two Euler equations solutions were obtained for a Mach number of 0.76 and an angle of attack of 2.5 deg. Two grids were used. The first, a

105- by 70-point 0-mesh, provided a reference solution. The second consisted of a two-level embedded grid; the first level, a 105- by 28-point 0-mesh, the second, a 105- by 46-point 0-mesh (Fig. 9). All grids had a 5-point overlap at the trailing edge cut so that the effective number of points defining the airfoil was reduced to 100. The elliptic grid generator GRAPE (Ref. 52) was used to construct all grids. The reference and chimera grids had the same point distribution on the airfoil and along the far-field boundary.

Both solutions were obtained with a global Courant number of 10 as defined by the smallest cell volume. Thus, the time steps between the component grids in the chimera scheme were different by a factor of 50. An ancillary result of the chimera solution was an improvement in convergence rate over the single-mesh solution by a factor of 3. The mismatch in the solution-pressure coefficients was discussed in Section 3.3. The major point to be noted is that the shock location, and hence the surface pressure distribution, is affected by the shock/grid-boundary interaction.

5.1.3 Maneuver Flap

Figure 11 illustrates the maneuver flap geometry that was created from the basic SKF1.1 geometry (Ref. 51). Some liberties were taken with the geometry for purposes of mesh generation. The abrupt change in curvature at the flap location on the lower surface of the airfoil was reduced by arbitrarily rounding the corner; the finite thickness of the trailing edge of the airfoil at the flap location was eliminated in favor of a sharp trailing edge; and the flap retained the sharp trailing edge discussed in Section 5.1.2.

A three-level grid hierarchy was employed. The first-level grid, G_{11}, consisted of a 105- by 28-point 0-mesh; the second-level mesh, G_{21}, was a 105- by 46-point 0-mesh which contained the airfoil; and the third-level grid, G_{31}, consisted of a 55- by 16-point 0-mesh which contained the flap. Each grid had a 5-point overlap at the cut to avoid the use of an implicit periodic boundary condition. Grids G_{11} and G_{21} are shown in Fig. 12 without G_{31}; G_{31} is shown embedded in G_{21} in Fig. 13. The grids were combined so that the flap chord line formed a 10-deg angle, β, with the chord line of the airfoil. The flap gap indicated in Fig. 13 is approximatly 1.5 times larger than the experimental configuration. The larger gap greatly simplified construction of the flap mesh G_{31} because points of G_{31} were constrained to lie completely within the computational domain (i.e. outside the airfoil).

Two solutions were obtained for supercritical conditions, $M_\infty = 0.6$, $\alpha = 3$ deg, $\beta = 10$ deg, and $M = 0.7$, $\alpha = 3$ deg, $\beta = 10$ deg. The angle of attack, α, is measured relative to the airfoil chord. Figures 14 and 15 present a comparison of the computations with experimental data (Ref. 51). Agreement is better for the $M_\infty = 0.6$ condition although the pressure peaks are overpredicted on both the airfoil and the flap. The overprediction of the suction peak causes a stronger shock which induces a downstream re-expansion. The effects of the modified lower surface geometry at about 70-percent chord are also visible. The $M_\infty = 0.7$ solution also overpredicts the expansion on the suction surface resulting in a stronger shock which is too far aft. The shock location at the trailing edge of the airfoil produces a higher pressure over the flap and reduces the predicted flap suction peak. The major cause of the disagreement with experiment is the lack of viscous effects in the computations which are known to be significant for supercritical airfoils. Nevertheless, the solutions demonstrate the ability of the method to simplify grid generation for complex geometries.

The Mach number contours for the two solutions are presented in Figs. 16 and 17. The $M_\infty = 0.6$ condition (Fig. 16) Mach number contours pass smoothly through the grid boundaries. The shock is weak and does not cross the grid boundary. The $M_\infty = 0.7$ Mach number contours (Fig. 17) show significant distortion of the shock at the grid boundaries. The resultant shock distortion at the grid interfaces is stronger for $M_\infty = 0.7$ solution than for the $M_\infty = 0.6$ solution (See Section 3.3).

5.1.4 Conclusions Drawn from 2-D Work

The most important conclusion drawn from the 2-D work just described is that the chimera scheme is a viable technique for simplification of grid generation. The procedures for combining grids, locating overlap boundaries, constructing holes, identifying interpolation points, and manipulating complex data structures were demonstrated and found to be workable.

Several problem areas were also identified. The restriction which limited hole formation to embedded grids did not provide sufficient flexibility. The algorithm for constructing holes allowed the embedded grid $G_{\ell+1,j}$ to induce a hole in mesh $G_{\ell,i}$ in which it was embedded; however, $G_{\ell,i}$ could not cause a similar hole to be formed in $G_{\ell+1,j}$. This problem became evident with the construction of the flap mesh described in Section 5.1.3. The outer boundary of the flap mesh had to be distorted so that it would not intersect the airfoil. As a result, a compromise was reached in which the gap between the airfoil and flap was increased beyond that of the experiment so that a flap mesh with a reasonable outer boundary could be constructed. This feature became prohibitive for the moving mesh computations described in Ref. 21. For the computations of Ref. 21 this restriction on hole formation was removed.

As has been noted in Section 3.3, the shock/grid-boundary interaction was found to be much more troublesome than originally expected. The problem grew as the shock strength increased. It was speculated that the nonconservative interpolation could be the cause of the difficulties. Although much effort has been devoted to verifying this hypothesis, no definitive answer has been obtained. Some significant findings have come to light; the use of nonreflecting boundary conditions reduces the interaction (Ref. 33); the type of interpolation scheme and extent of the overlap region are important (Refs. 18, 32, 33, 34, and 35); and the relative difference in spatial resolution may also be very important (e.g., Refs. 18 and 35).

The convergence rates of the solutions given in Sections 5.1.2 and 5.1.3 were slow. While the convergence rates were not of primary concern, it became obvious that something must be done to make 3-D computations feasible. In Section 5.1.1, improvements in convergence rates were observed on the chimera grids. This result suggested that the different time steps on each grid were important for further accelerations. Therefore, the 3-D extension used local time stepping everywhere.

5.2 3-D APPLICATIONS

The 3-D capability of the chimera grid embedding is demonstrated by three example configurations. The first example is a generic three-body configuration consisting of three ellipsoids in a triangular arrangement. The second two are closely related wing/body and a wing/body/tail combination. The embedding hierarchy used in these examples is shown in Fig. 18. As in the 2-D case, the hierarchy requires that embedded grids, $G_{\ell+1,j}$, $j = 1, 2, \ldots,$ be contained completely within the enveloping mesh, $G_{\ell,i}$. The examples illustrate the wide range of geometries that may be accommodated within the simple hierarchical framework. The wing/body and wing/body/tail configurations illustrate the manner in which a complex geometry can be represented by adding component grids. All of the following solutions were obtained on the AEDC Cray Model XMP 1/2 computer.

5.2.1 Three-Body Configuration

A generic, three-body configuration was considered to test the flexibility of the embedding hierarchy. The configuration consists of three ellipsoidal bodies in a triangular arrangement (Fig. 19). The grids of the two smaller bodies have major and minor axes one-half of the larger; each ellipsoid has a length-to-maximum-diameter ratio of 10. The two smaller bodies are embedded in the mesh of the larger as indicated in Fig. 19. All the grids are spherical with a 5-point overlap in the η coordinate. The mesh of the large ellipsoid has 26,250 points distributed 30 by 35 by 25 in the ξ, η, and ζ directions; the meshes of the two smaller ellipsoids have 15,750 points distributed 30 by 35 by 15; and the composite mesh has 57,750 points. All grids were constructed using a hyperbolic grid generator (Refs. 53 and 54).

5.2.2 Computation of Three-Body Configuration

A computation was made for $M_\infty = 0.8$ and $\alpha = -2.0$ deg using the complete mesh with no assumptions of symmetry. Thus, any asymmetries observed in the solution would be a result of an artificial asymmetry built into the calculation procedure. Surface contours of Mach numbers are shown in Fig. 20. The contours indicate that the flow between the bodies is symmetrical. Therefore, it was concluded that the codes were functioning properly.

5.2.3 Wing/Body

The wing/body configuration was designed to provide a simple configuration which could be used to assess wind tunnel wall interference (Ref. 55). It consists of a blunted ogive-cylinder fuselage and a midmounted wing (Fig. 21). The wing has a constant chord planform which is swept back at 30 deg with no twist or taper. Cross sections of the wing parallel to the plane of symmetry are NACA-0012 airfoils. The wing/body dimensions in units of fuselage cylinder radii are shown in Fig. 21. The figure includes the dimensions of the tail which has a constant chord planform swept back at 30 deg without twist or taper. The equations describing the fuselage geometry are

$$y = \begin{cases} \sqrt{(0.427 - x)x} & 0 \le x \le 2.42 \\ 0.162 - 0.286x - 0.024x^2 & 2.42 < x \le 4.11 \\ 1.0 & 4.11 < x \end{cases} \tag{16}$$

The dimensionless model coordinates x and y are indicated in Fig. 21. The model has been tested in several wind tunnels over a wide range of Mach and Reynolds numbers; however, the experimental data are as yet unpublished. An assessment of their accuracy is underway.

5.2.4 Grids

The 3-D grid-embedding process is illustrated with the wing/body configuration (See Fig. 18). An outer mesh, Fig. 22, encloses the model. It is a warped, hemispherical shell whose polar axis is coincident with the fuselage centerline. The mesh was constructed by using the GRAPE code (Ref. 52) to generate a mesh in a longitudinal plane. The plane was then rotated about the polar axis. The mesh extends from 9 to 51 radii from the fuselage and contains 19,740 points which are distributed 47 by 21 by 20 in the ξ, η, and ζ directions (See Fig. 22). The fuselage mesh (Fig. 23) is also a warped hemispherical shell whose inner boundary is the fuselage surface. The grid contains 29,375 points distributed 47 by 25 by 25 and extends to 11.5 radii. Thus, the outer boundary of the fuselage mesh overlaps the inner boundary of the outer mesh by 2.5 radii (about 4- to 5-point overlap). The wing grid (Fig. 24) is a warped cylindrical mesh whose axis is directed along the wing span. The end surface containing the wing root is coincident with the fuselage surface. The grid was constructed by using the GRAPE code (Ref. 52) to generate 0-mesh grids at selected spanwise planes. The planar grids were then sheared onto cylindrical surfaces whose radius was equal to the spanwise location. The mesh contains 16,698 points distributed 66 by 23 by 11. The ξ coordinate (Fig. 24) contains a 5-point overlap at the trailing edge cut to eliminate the requirement for an implicit periodic solution; the η coordinate has 15 spanwise surfaces defining the wing. The composite mesh has a total of 65,813 points.

Because the fuselage mesh has points which lie within the wing, points in the neighborhood of the wing are removed from the fuselage grid. Figure 25 displays the resulting hole boundary. Values of the dependent variables at points on the hole surface are obtained from the wing mesh by trilinear interpolation (See Section 3.3 and Appendix B).

5.2.5 Wing/Body Computations

Three calculations of the flow about the wing/body were made, (1) a subcritical, compressible flow at $M_\infty = 0.6$ and $\alpha = 0$ deg, (2) a slightly supercritical flow at $M_\infty = 0.75$ and $\alpha = 4$ deg, and (3) a highly supercritical flow at $M_\infty = 0.9$ and $\alpha = 2$ deg. In the comparisons of computed and experimental data that follow, the pressure coefficient, C_p, is plotted as a function of the local dimensionless chord, X/C, where X is aligned in planes parallel to the plane of symmetry. Experimental data are available at three spanwise locations. In terms of the fraction of the semispan, Y/(b/2), the locations are 0.4, 0.6, and 0.9. Data are also available on the fuselage upper surface in the symmetry plane and are presented as a funtion of the dimensionless fuselage length, X/D, where D = 10 in. This scale was chosen to facilitate plotting. Note that the computational model continued the cylindrical portion of the fuselage to X/D = 1.4, whereas the experimental model ended at X/D = 1.2. No effort was made to model the support structure.

The subcritical condition, $M_\infty = 0.6$ and $\alpha = 0$ deg, was selected as an initial test of the 3-D chimera technique. Figure 26 presents a comparison of computed and experimental (Ref. 55) pressure coefficients. The agreement is favorable, even near the tip. No effort was made to model the tip; the only computational requirement was that the grid be packed somewhat near the tip. Packing was achieved using hyperbolic tangent spacing obtained from the method described in Ref. 56. The comparison with the fuselage data is good except in the region where the tail is located ($X/D \cong 1.0$). In that region, the computation predicts a constant value of $C_p \cong 0$, whereas the data show the flow to be slightly accelerated.

The slightly supercritical condition, $M_\infty = 0.75$ and $\alpha = 4.0$ deg, was investigated next. A comparison of the computed and experimental pressure data is made in Fig. 27. Again the comparison is encouraging. The computation overpredicts the suction pressure on the wing upper surface, and the disagreement increases toward the tip. Much of this disparity is attributable to the increasing importance of viscous effects as the flow becomes supercritical. The fuselage data continue to be well predicted except near the tail location where the disagreement is larger than in the subcritical case.

The supercritical condition was computed for $M_\infty = 0.9$ and $\alpha = 2$ deg. Figure 28 compares the computation with the experimental data. The agreement is acceptable. The agreement near the wing tip remains surprisingly good; the disagreement in the region of the tail has become much more significant. The fuselage data show the presence of a shock wave slightly downstream of the wing trailing edge. The computed shock surface is shown in red in Fig. 29. The shock extends to the symmetry plane from a complex shock structure at the wing/fuselage junction. The ragged nature of the shock surface is caused by the plot program (Ref. 57). Figure 29a shows the curved structure of the shock from the root to the tip (See Fig. 28). Because the shock is ''painted'' last by the plot program, the lower surface shock also appears in Fig. 29a as the most forward patch of red near the wing tip. Figure 29b shows the lower surface shock more clearly. The shock location on the wing/body surface can also be seen in Fig. 30 which displays the surface grids of the fuselage and wing; the portion of the wing grid that is coincident with the fuselage is also shown. Mach number contours on the body surface show the $M = 1.0$ (green) contour from the symmetry plane down the fuselage to the wing root and across the wing. The figure indicates that a major portion of the upper wing surface is supersonic (i.e. region between the green contours). The expansion over the wing is sufficiently strong to induce a supercritical flow on the fuselage. Mach number contours in η = constant surfaces at the wing root, midspan, and tip are presented in Fig. 31. The dotted lines in the figure represent the computational mesh and the solid lines are the Mach number contours. The shock (green, $M = 1.0$ contour) is smeared because of insufficient clustering of grid points. The contours at the grid boundaries are as smooth as the spatial resolution allows.

Figure 32 displays Mach number contours on the outer boundary of the wing mesh. These contours are of interest as they result from interpolations in the fuselage grid. The exchange of information between the grids results in a smooth set of contours. The sonic bubble on the wing (green contour) passes through the outer boundary. The shock surface (Fig. 29b) continues into the fuselage mesh where the differences in spatial resolution between the grids smear the shock.

5.2.6 Wing/Body/Tail Configuration

The horizontal tail was added to the wing/body and new grids were constructed using the techniques described in Section 5.2.4. The outer grid has 37,000 points (74 by 25 by 20); the fuselage mesh contains 77,700 (74 by 35 by 30); the wing mesh has 27,720 points (66 by 28 by 15) with 20 points in the η direction defining the wing surface; and the tail contains 15,120 points (56 by 18 by 15) with 10 points in the η direction defining the tail surface. Thus, the composite grid consists of four component grids and has 157,540 points. The new grids were used to test the behavior of the chimera scheme with large component grids.

Because the fuselage mesh has points which lie within the wing and tail, two holes are introduced into the fuselage grid in the neighborhood of the wing and tail. Figure 33 displays the resulting hole boundaries. Values of the dependent variables on the hole surfaces must be interpolated from either the wing or tail grids, as appropriate (See Section 3.3 and Appendix B).

5.2.7 Wing/Body/Tail Computations

The M_∞ and $\alpha = 2.0$ deg condition was rerun for the complete configuration. A comparison of experimental and computed pressure coefficient, C_p, as a function of the dimensionless chord X/C is presented in Fig. 34 in the same manner as Section 5.2.5. Data are available for only one semispan location on the tail, $Y/(b_t/2) = 0.60$. Figure 34 shows the computed results to be in excellent agreement with the experimental data. The addition of the tail had very little effect on the wing pressure distributions. The agreement with the fuselage data is significantly improved. However, the data show a slightly more extensive expansion on the fuselage than is computed. The tail data and the computation indicate that the 2-deg angle of attack is negated by the downwash from the wing. The data indicate the presence of a shock which is not observed in the calculation. Comparison of the solution for the wing/body configuration presented in Fig. 28 with that in Fig. 34 shows some discrepancies which are attributed to differences in spatial resolution and convergence between the solutions. The large composite grid of the wing/body/tail (157,540 points) was not converged to the same degree as the wing/body grid (65,813 points), a three order-of-magnitude reduction of the residual compared to four.

The fuselage data (Fig. 34) indicate the presence of a shock wave (See Section 5.2.5). The computed shock wave structure is shown in red in Fig. 35. A shock wave extends from the fuselage symmetry plane around the fuselage to the wing root, across the upper surface of the wing to the wing tip, and around the tip to the lower surface as in Fig. 29. The shock in Fig. 35 is sharper and less ragged because of the increased spatial resolution. A small shock can also be seen on the tail. This shock is weaker and does not extend to the tail root nor does it cross the tail grid outer boundary. This is consistent with the effective reduction of the angle of attack at the tail noted in Fig. 34. Mach number contours on the full configuration are shown in Fig. 36, which also displays the surface grids (compare with Fig. 30). The Mach = 1.0 (green) contours can be traced around the fuselage and across the wing. A large portion of the wing upper surface is supercritical. In comparison, only a small region concentrated near the tip is supercritical on the tail. Figure 37 presents Mach number contours in η = constant surfaces at the root, midspan, and tip for both the wing and tail. The small extent of the supercritical flow on the tail is evident (the green, M = 1.0 contour).

Figure 38 displays Mach number contours on the outer boundaries of the wing and tail grids which result from quantities interpolated from the fuselage mesh. The information exchange among the grids results in smooth contours. The sonic bubble over the wing (green contour) passes through the grid boundary, whereas the tail has no such interaction.

6.0 CONCLUDING REMARKS

A set of computer codes have been described that implement 3-D grid-embedding techniques as a part of a flexible solution concept that we have called a chimera method. The codes utilize procedures for combining grids, locating embedded boundaries and interpolation points, and manipulating complex data structures. The validity of the method was successfully demonstrated on several geometries for inviscid flow. Extension of the method to include viscous effects is underway.

REFERENCES

1. "Numerical Grid Generation Techniques." NASA CP 2166, Workshop held at NASA Langley Research Center, Hampton, Virginia, October 6-7, 1980.

2. Thompson, J. F. and Warsi, Z.U.A. "Boundary-Fitted Coordinate System for Numerical Solution of Partial Differential Equations: A Review." *Journal of Computational Physics*, Vol. 47, No. 1, 1982, pp. 1–108.

3. Thompson, J. F., ed. *Numerical Grid Generation*. Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Their Use in the Numerical Solution of Partial Differential Equations, Held in Nashville, Tennessee, April 1982, North-Holland, New York, NY, 1982.

4. Ghia, K. N. and Ghia, U., eds. *Advances in Grid Generation*, Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Sponsored by The Fluids Engineering Division of ASME, FED–Vol. 5, Held in Houston, Texas, June 1983.

5. Rubbert, P. E. and Lee, K. D. "Patched Coordinate Systems." *Numerical Grid Generation*, J. F. Thompson, ed., North-Holland, New York, New York, 1982, pp. 235–252.

6. Lee, K. D. "3-D Transonic Flow Computations Using Grid Systems with Block Structure." AIAA Paper No. 81-0998, June 1981.

7. Lasinski, T. A. et al. "Computation of the Steady Viscous Flow Over a Tri-Element 'Augmentor Wing' Airfoil." AIAA Paper No. 82-0021, January 1982.

8. Hessenius, K. A. and Pulliam, T. H. "A Zonal Approach to Solutions of the Euler Equations." AIAA Paper No. 82-0969, June 1982.

9. Rai, M. M. "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations." AIAA Paper No. 84-0164, January 1984.

10. Miki, K. and Takagi, T. "A Domain Decomposition and Overlapping Method for the Generation of Three-Dimensional Boundary-Fitted Coordinate Systems." *Journal of Computational Physics*, Vol. 53, No. 2, February 1984, pp. 319–330.

11. Holst, T. L. et al. "Numerical Solution of Transonic Wing Flows Using an Euler/Navier-Stokes Zonal Approach." AIAA Paper No. 85-1640, July 1985.

12. Atta, E. H. "Component-Adaptive Grid Interfacing." AIAA Paper No. 81-0382, January 1981.

13. Atta, E. H. and Vadyak, J. A. "A Grid Interfacing Zonal Algorithm for Three-Dimensional Transonic Flows About Aircraft Configurations." AIAA Paper No. 82-1017, June 1982.

14. Steger, J. L., Dougherty, F. C., and Benek, J. A. "A Chimera Grid Scheme." *Advances in Grid Generation*, K. N. Ghia and U. Ghia, eds., ASME FED-Vol. 5, June 1983.

15. Benek, J. A., Steger, J. L., and Dougherty, F. C. "A Flexible Grid Embedding Technique with Application to the Euler Equations." AIAA Paper No. 83-1944, July 1983.

16. Benek, J. A., Buning, P. G., and Steger, J. L. "A 3-D Chimera Grid Embedding Technique." AIAA Paper No. 85-1523, July 1985.

17. Lombard, C. K. and Venkatapathy, E. "Implicit Boundary Treatment for Joined and Disjoint Patched Mesh Systems." AIAA Paper No. 85-1503, July 1985.

18. Venkatapathy, E. and Lombard, C. K. "Flow Structure Capturing on Overset Patch Meshes." AIAA Paper No. 85-1690, July 1985.

19. Fuchs, L. "Numerical Flow Simulation Using Zonal-Grids." AIAA Paper No. 85-1518, July 1985.

20. Rai, M. M. "Navier-Stokes Simulations of Rotor-Stator Interaction Using Patched and Overlaid Grids." AIAA No. 85-1519, July 1985.

21. Dougherty, F. C. "Development of a Chimera Grid Scheme with Applications to Unsteady Problems." Ph.D Dissertation, Stanford Univrsity, April 1985.

22. Dougherty, F. C., Benek, J. A., and Steger, J. L. "On Applications of Chimera Grid Schemes to Store Separation." NASA TM88193, October 1985.

23. Wedan, B. and South, J. C., Jr. "A Method for Solving the Transonic Full-Potential Equation for General Configurations." AIAA Paper No. 83-1889, July 1983.

24. Gnoffo, P. A. "A Vectorized, Finite Volume, Adaptive Grid Algorithm for Navier-Stokes." *Numerical Grid Generation*, J. F. Thompson, ed., North-Holland, New York, New York, 1982, pp. 819–836.

25. Nakahashi, K. and Deiwert, G. S. "A Self-Adaptive-Grid Method with Application to Airfoil Flow." AIAA Paper No. 85-1525, July 1985.

26. Ghia, K., Ghia, U., and Shin, C. T. "Adaptive Grid Generation for Flows with Local High Gradient Regions." *Advances in Grid Generation*, K. N. Ghia and U. Ghia, eds., ASME-FED-Vol. 5, June 1983.

27. Brackbill, J. V. "Coordinate System Control: Adaptive Meshes." *Numerical Grid Generation*, J. F. Thompson, ed., North-Holland, New York, New York, 1982, pp. 277–294.

28. Saltzman, J. and Brackbill, J. V. "Application and Generation of Variational Methods for Generating Adaptive Systems." *Numerical Grid Generation*, J. F. Thompson, ed., North-Holland, New York, New York, 1982, pp. 865–884.

29. Berger, M. J. "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations." Stanford University STAN-CS-82-924, August 1982.

30. Berger, M. J. and Oliger, J. "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations." *Journal of Computational Physics*, Vol. 53 No. 3, March 1984, pp. 484–512.

31. Chaderjian, N. M. and Steger, J. L. "A Zonal Approach for the Steady Transonic Simulation of Inviscid Rotational Flow." AIAA Paper No. 83-1927, July 1983.

32. Berger, M. J. "On Conservation at Grid Interfaces." NASA Langley Research Center ICASE Report 84-43, September 1984.

33. Eberhardt, S. "Overset Grids in Compressible Flow." AIAA Paper No. 85-1524, July 1985.

34. Mastin, C. W. and McConnaughey, H. V. "Computational Problems on Composite Grids." AIAA Paper No. 84-1611, June 1984.

35. Parthasarathy, K. N. "Numerical Procedure for Aircraft/Store Flow Field." Quarterly Progress Reports Nos. 4, 5, and 6, prepared under NASA Contract NAS2-11742 by General Dynamics, Fort Worth Division, Fort Worth, Texas, 1984.

36. Knuth, D. E. *The Art of Computer Programming, Vol. 1–Fundamental Algorithms*, Addison-Wesley Publishing Company, Reading, Massachusettes, 1968.

37. Horowitz, E. and Sahni, S. *Fundamentals of Data Structures*. Computer Science Press, Inc., Rockville, Maryland, 1976.

38. Chartrand, G. *Introduction to Graph Theory*. Dover Publications, Inc., New York, New York, 1985.

39. Norton, R. J. G., Thompkins, W. T., and Haimes, R. "Implicit Finite Difference Schemes with Nonsimply Connected Grids—A Novel Approach." AIAA Paper No. 84-0003, January 1984.

40. Beam, R. and Warming, R. F. "An Implicit Finite Difference Algorithm for Hyperbolic Systems in Conservation Law Form." *Journal of Computational Physics*, Vol. 22, No. 1, September 1976, pp. 87–110.

41. Beam, R. and Warming, R. F. "An Implicit Factored Scheme for Compressible Navier-Stokes Equations." AIAA Paper No. 77-645, June 1977.

42. Steger, J. L. "Implicit Finite-Difference Simulation of Flow About Two-Dimensional Geometries." *AIAA Journal*, Vol. 16, No. 4, July 1978, pp. 679–686.

43. Pulliam, T. H. and Steger, J. L. "On Implicit Finite Difference Simulations of Three-Dimensional Flows." AIAA Paper No. 78-10, January 1978.

44. Jameson, A., Schmidt, W., and Turkel, E. "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping." AIAA Paper No. 81-1259, June 1981.

45. Srinivasan, G. R., Chyu, W. J., and Steger, J. L. "Computation of Simple Three-Dimensional Wing-Vortex Interaction in Transonic Flow." AIAA Paper No. 81-2106, June 1981.

46. Pulliam, T. H. "Euler and Thin Layer Navier-Stokes Codes: ARC2D and ARC3D." Notes for Computational Fluid Dynamics User's Workshop, The University of Tennessee Space Institute, March 1984.

47. Flores, J. "Convergence Acceleration for a Three-Dimensional Euler/Navier-Stokes Zonal Approach." AIAA Paper No. 85-1495, July 1985.

48. Pulliam, T. H. and Chaussee, D. S. "A Diagonal Form of an Implicit Approximate-Factorization Algorithm." *Journal of Computational Physics*, Vol. 39, No. 2, February 1981, pp. 347–363.

49. Jacocks, J. L. "Evaluation of Interference Effects on a Lifting Model in the AEDC/PWT 4-ft Transonic Tunnel." AEDC-TR-70-72 (AD-868290), April 1970.

50. Salas, M. D. "Recent Developments in Transonic Euler Flow Over a Circular Cylinder." *Journal of Math and Computers in Simulation*. Vol. 25, No. 3, June 1983, pp. 232-236.

51. Stanewski, E. and Thibert, J. A. "Airfoil SKF1.1 with Maneuver Flap." AGARD AR-138, "Experimental Data Base for Computer Program Assessment," May 1979.

52. Sorenson, R. L. "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equations." NASA TM 81198, May 1980.

53. Steger, J. L. and Chaussee, D. "Generation of Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations." *SIAM Journal on Scientific and Statistical Computing*, Vol. 1, December 1980, pp. 431–437.

54. Kinsey, D. W. and Barth, T. J. "Description of a Hyperbolic Grid Generating Procedure for Arbitrary Two-Dimensional Bodies." AFWAL TM 84-191-FIMM, July 1984.

55. Parker, R. L., Jr. and Erickson, J. C., Jr. "Status of Three-Dimensional Adaptive-Wall Test Section Development at AEDC." AIAA Paper No. 84-0624, March 1984.

56. Vinokur, M. "On One-Dimensional Stretching Functions for Finite-Difference Calculations." *Journal of Computational Physics*, Vol. 50, No. 2, May 1983, pp. 215–234.

57. Buning, P. G. and Steger, J. L. "Graphics and Flow Visualization in Computational Fluid Dynamics." AIAA Paper No. 85-1507, July 1985.

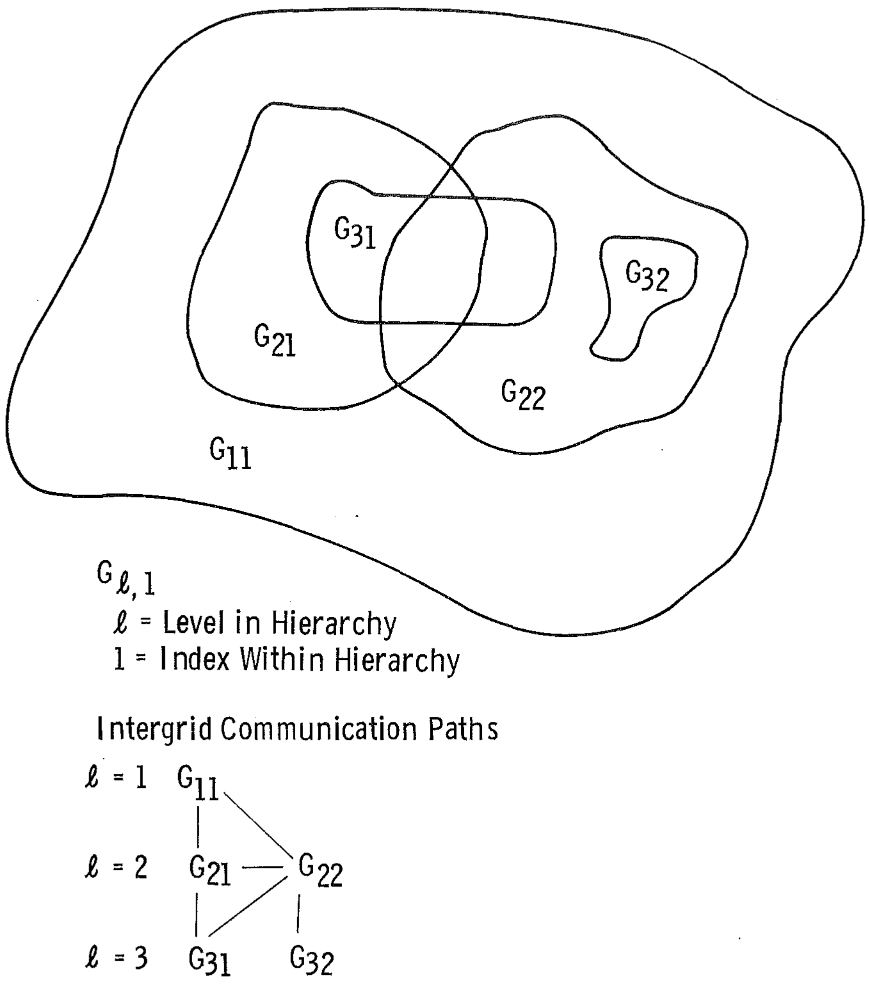

G_{31}
G_{32}
G_{21}
G_{22}
G_{11}
$G_{\ell,1}$
ℓ = Level in Hierarchy
1 = Index Within Hierarchy
Intergrid Communication Paths
$\ell = 1$ G_{11}
$\ell = 2$ G_{21} G_{22}
$\ell = 3$ G_{31} G_{32}

Figure 1. Hierarchical structure of embedded grids.

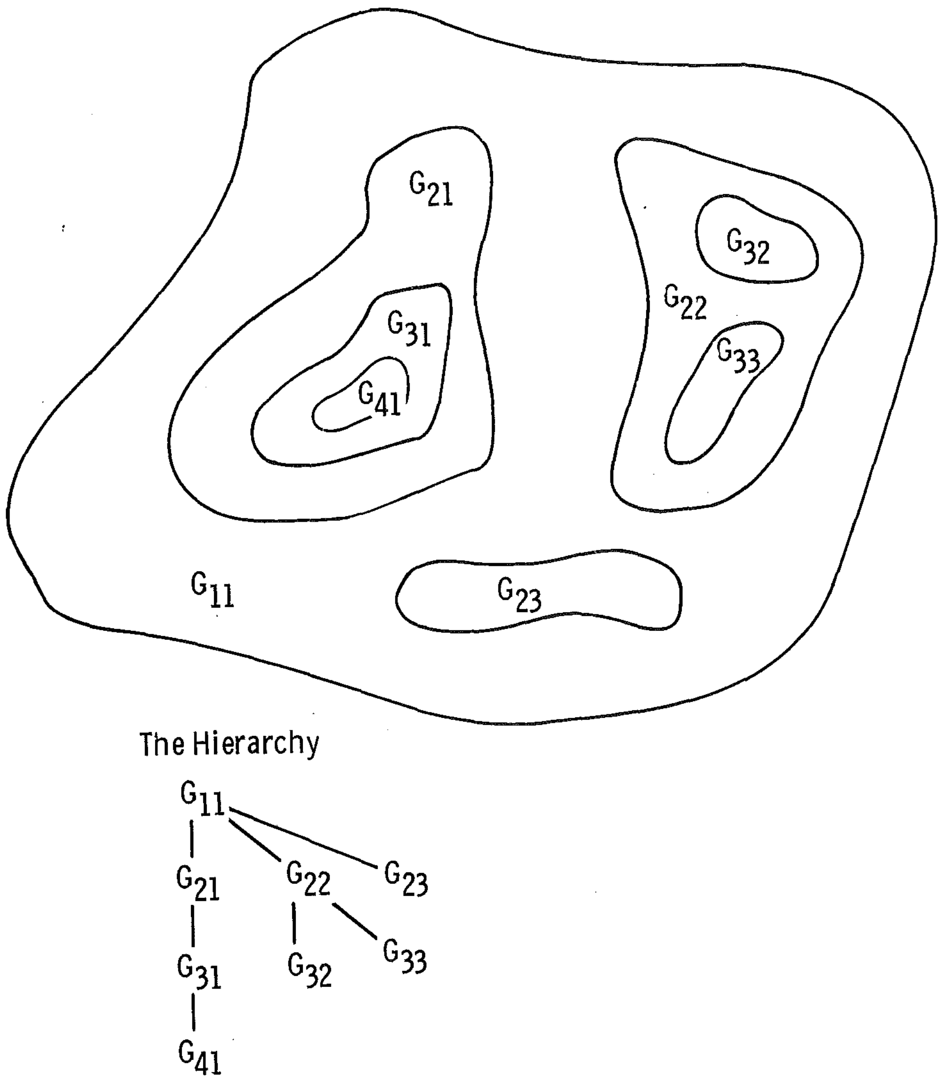


Figure 2. Restricted hierarchy used in this report.

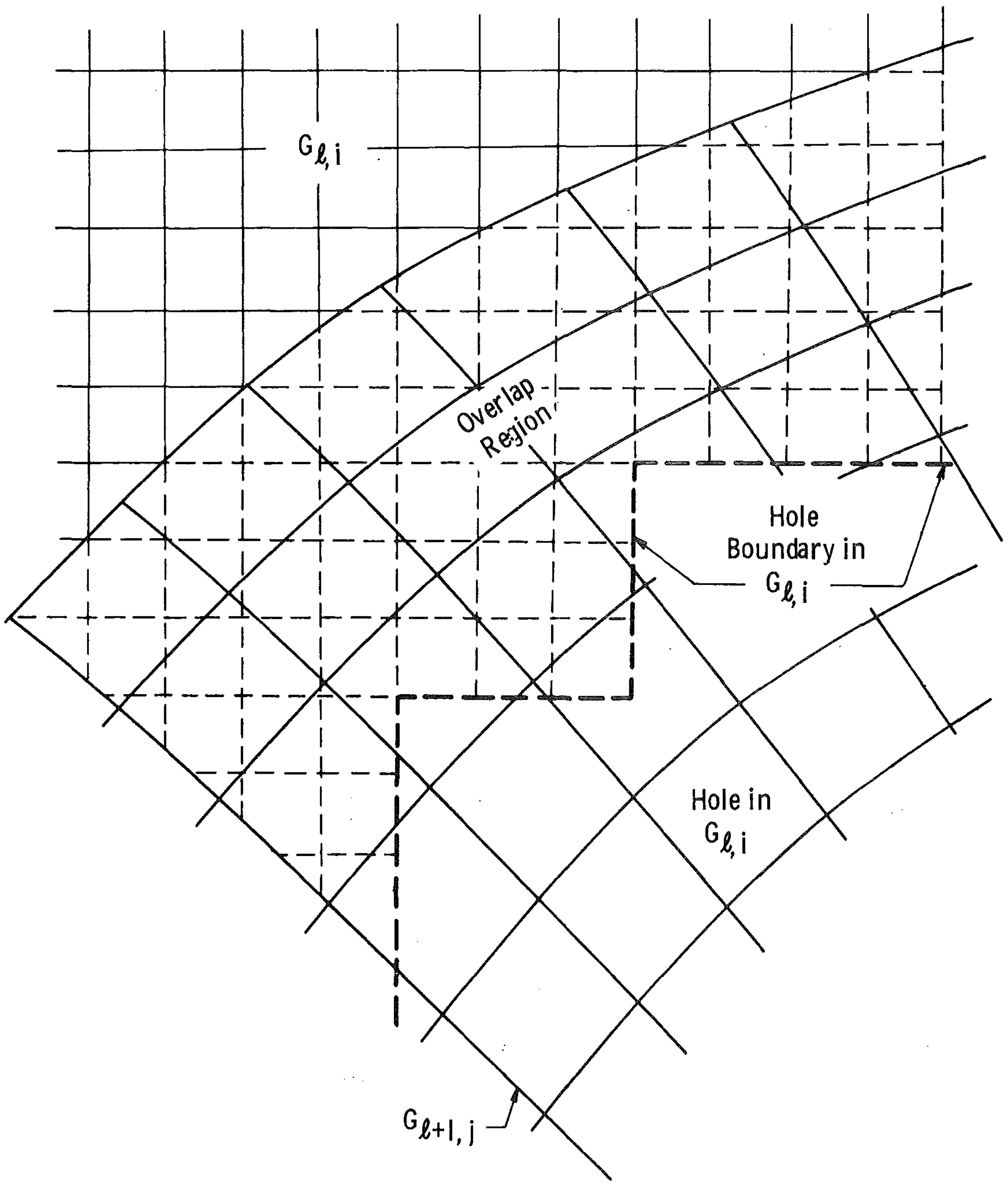
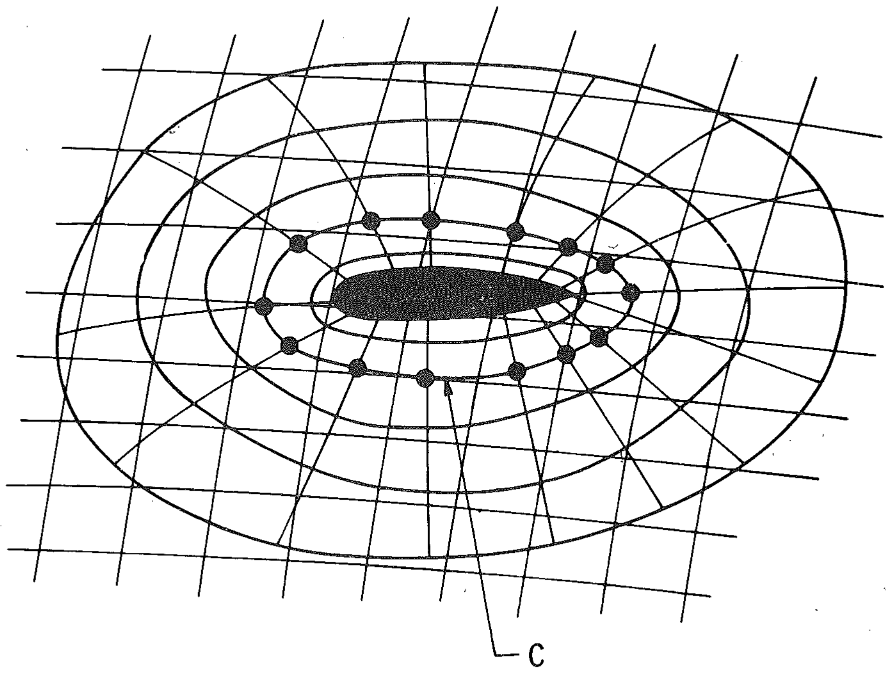
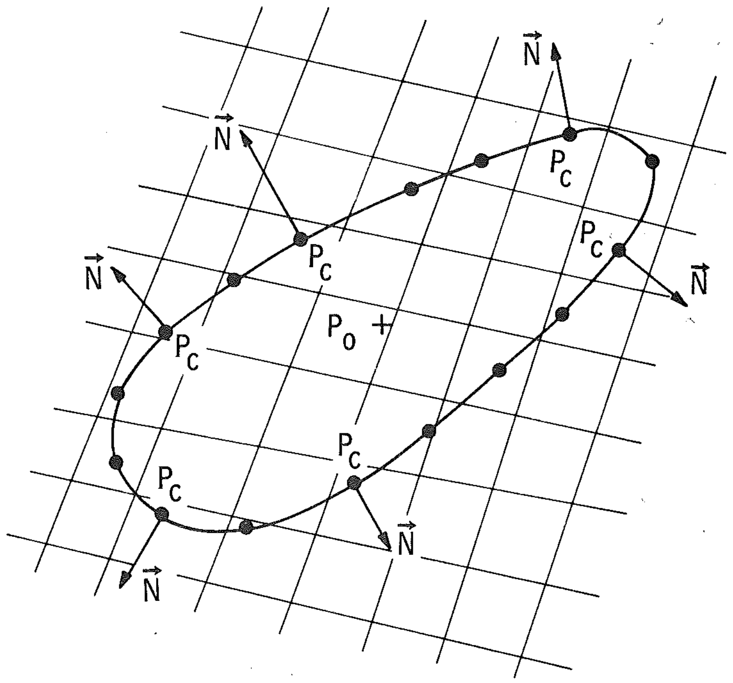


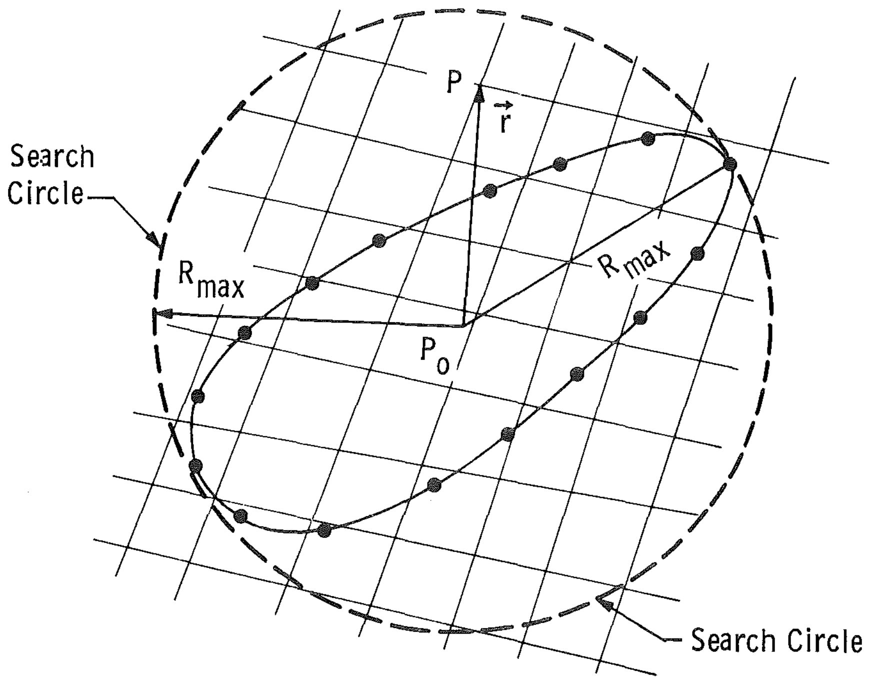
Figure 3. Overlap region between grids.



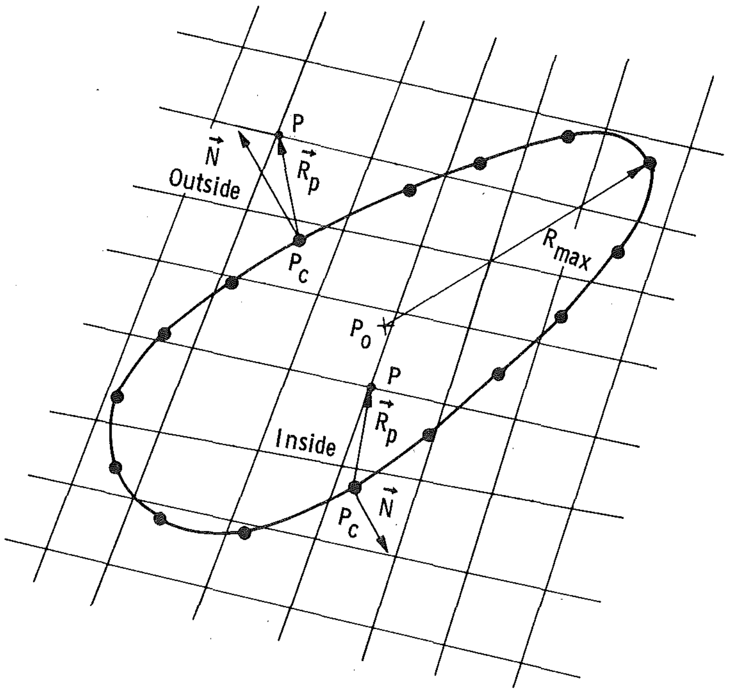
a. Initial hole boundary defined by level curve, C



b. Construction of outward normals to C
Figure 4. Illustration of hole construction in two dimensions.



c. Temporary origin (P_0) and construction of search circle



d. Construction of position vector $\vec{R}_p$ and dot product test
Figure 4. Concluded.

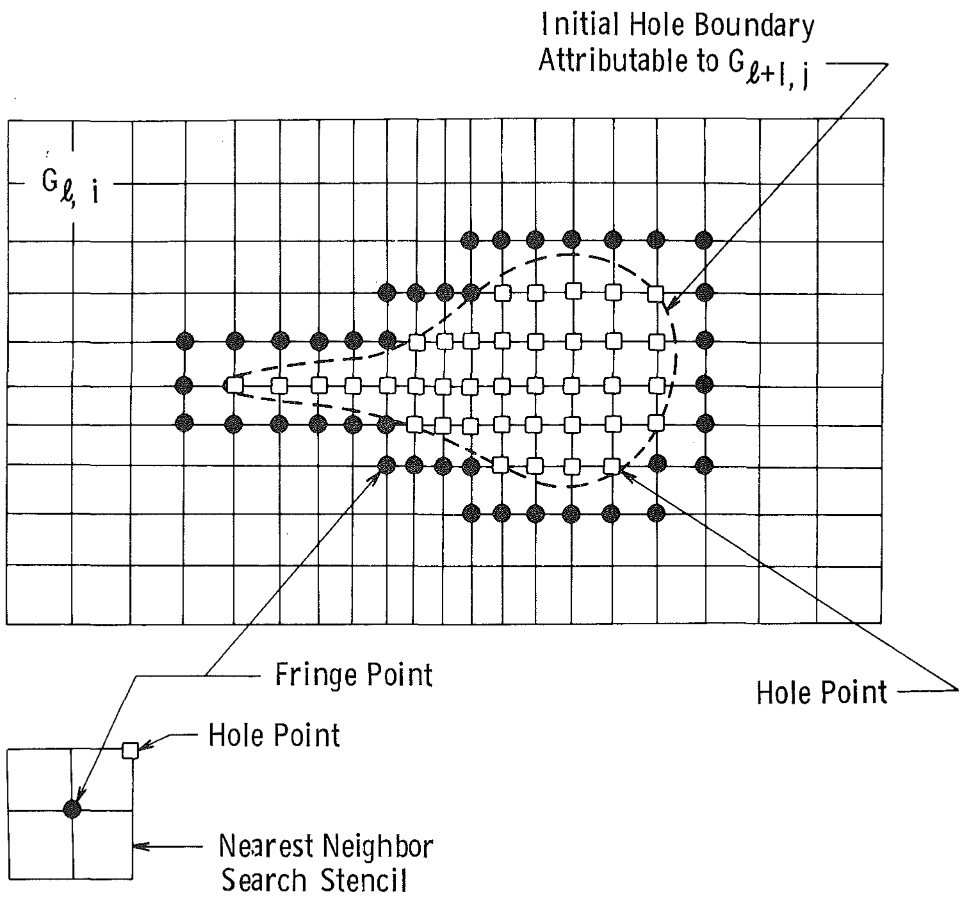


Figure 5. Final hole-boundary construction.

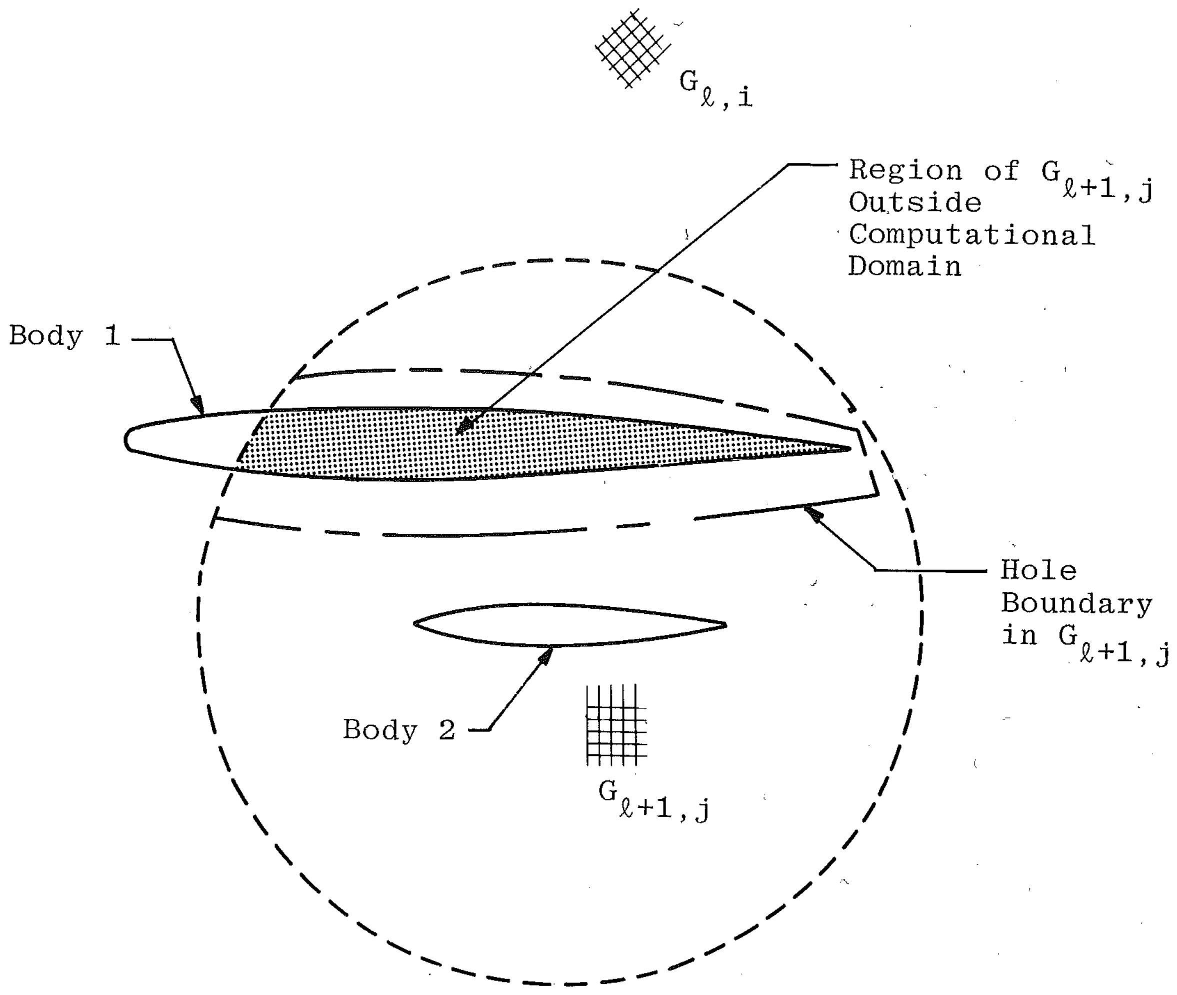

$G_{\ell,i}$
Region of $G_{\ell+1,j}$ Outside Computational Domain
Body 1
Hole Boundary in $G_{\ell+1,j}$
Body 2
$G_{\ell+1,j}$

Figure 6. Hole boundary in embedded grid $G_{\ell+1,j}$ caused by solid boundary in $G_{\ell,i}$.

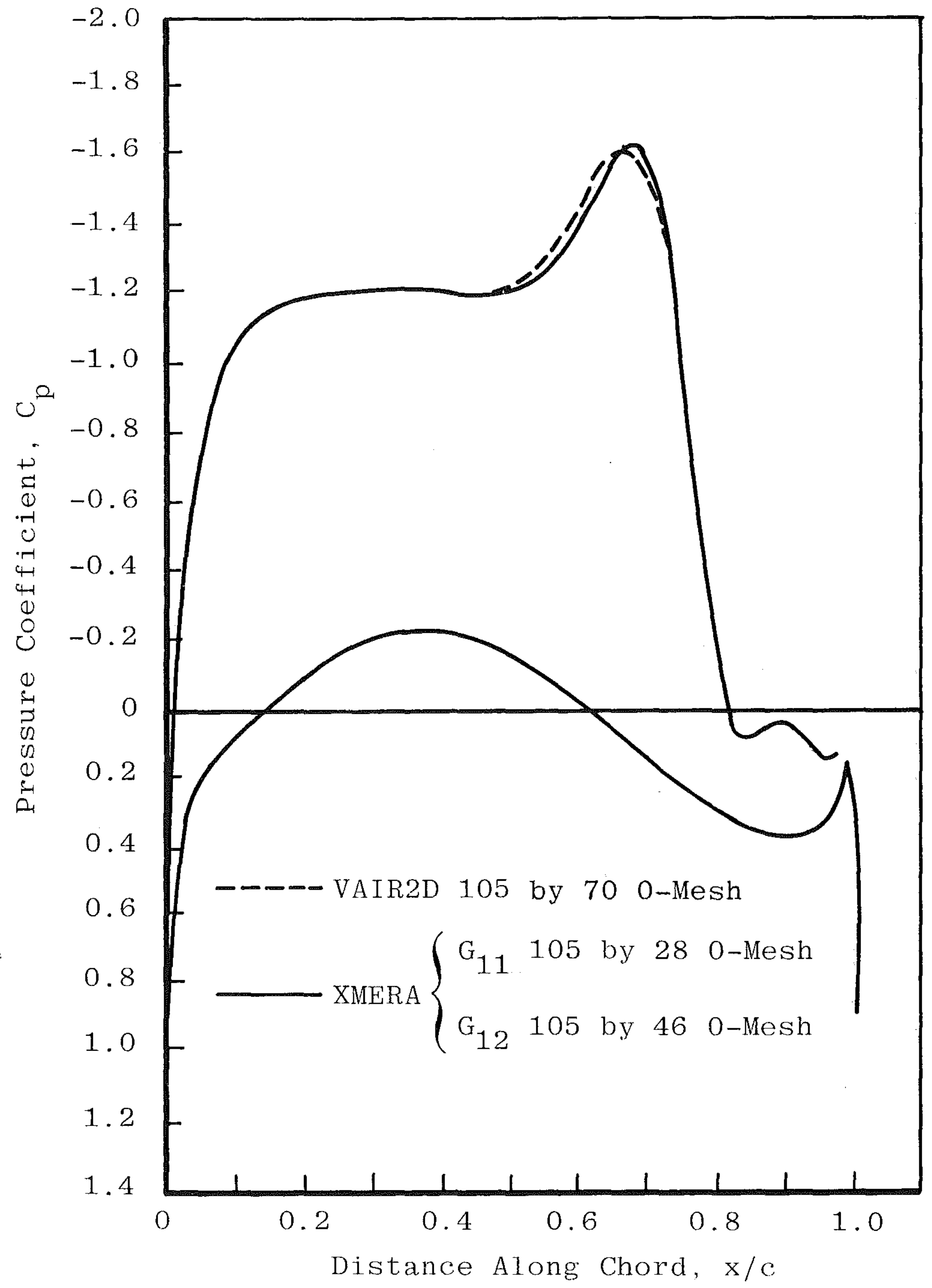


Figure 7. Comparison of single-grid and chimera-grid solutions for SKF1.1 airfoil geometry for supercritical conditions, $M_\infty = 0.76$, $\alpha = 2.5$ deg.

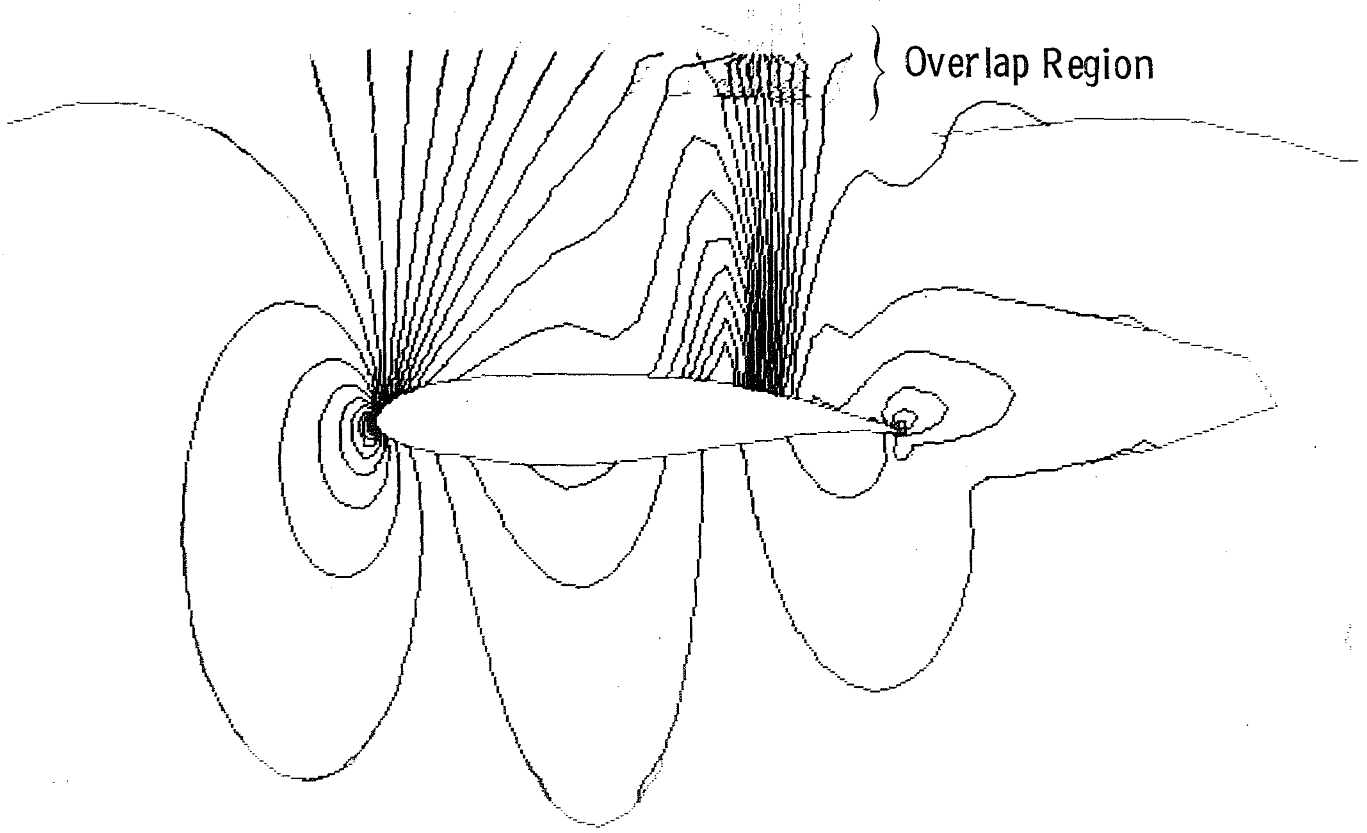


Figure 8. Mach number contours from chimera-grid solution, $M_\infty = 0.76$, $\alpha = 2.5$ deg.

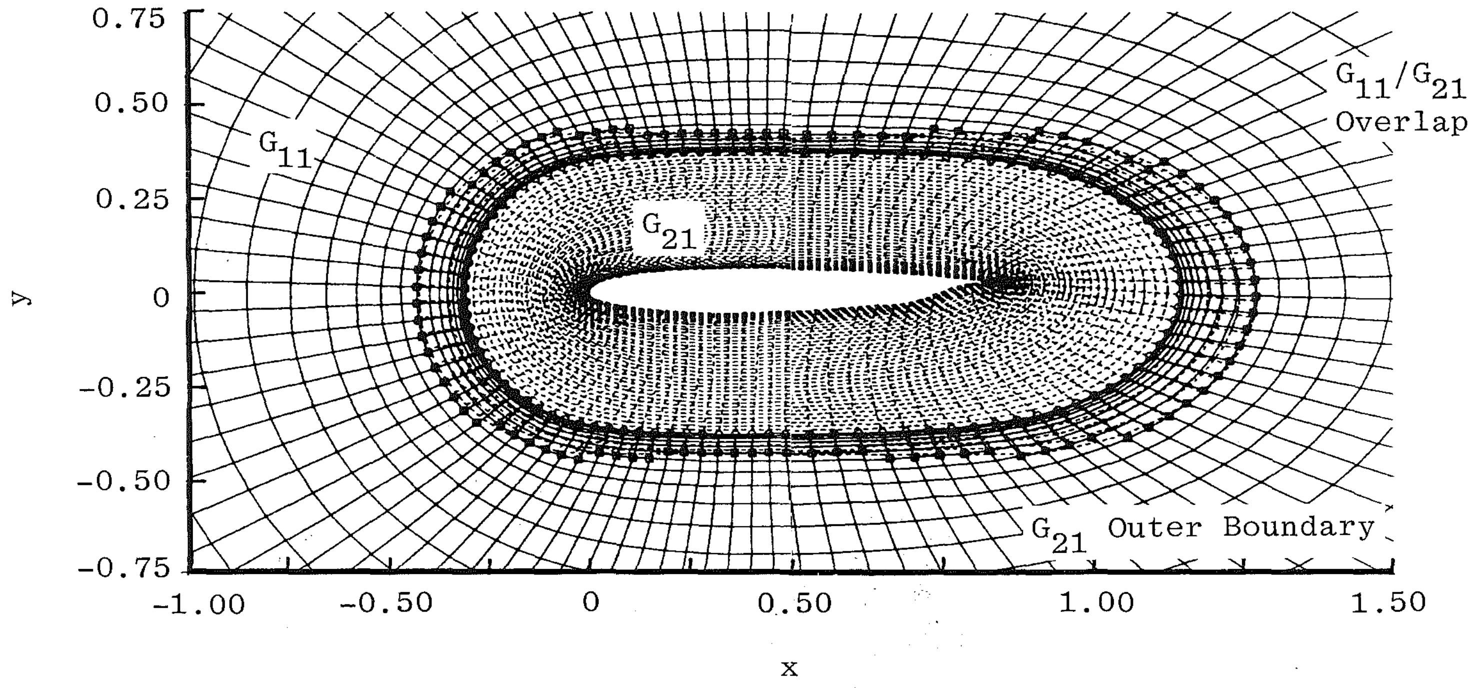


Figure 9. Chimera grid for the SKF1.1 Airfoil.

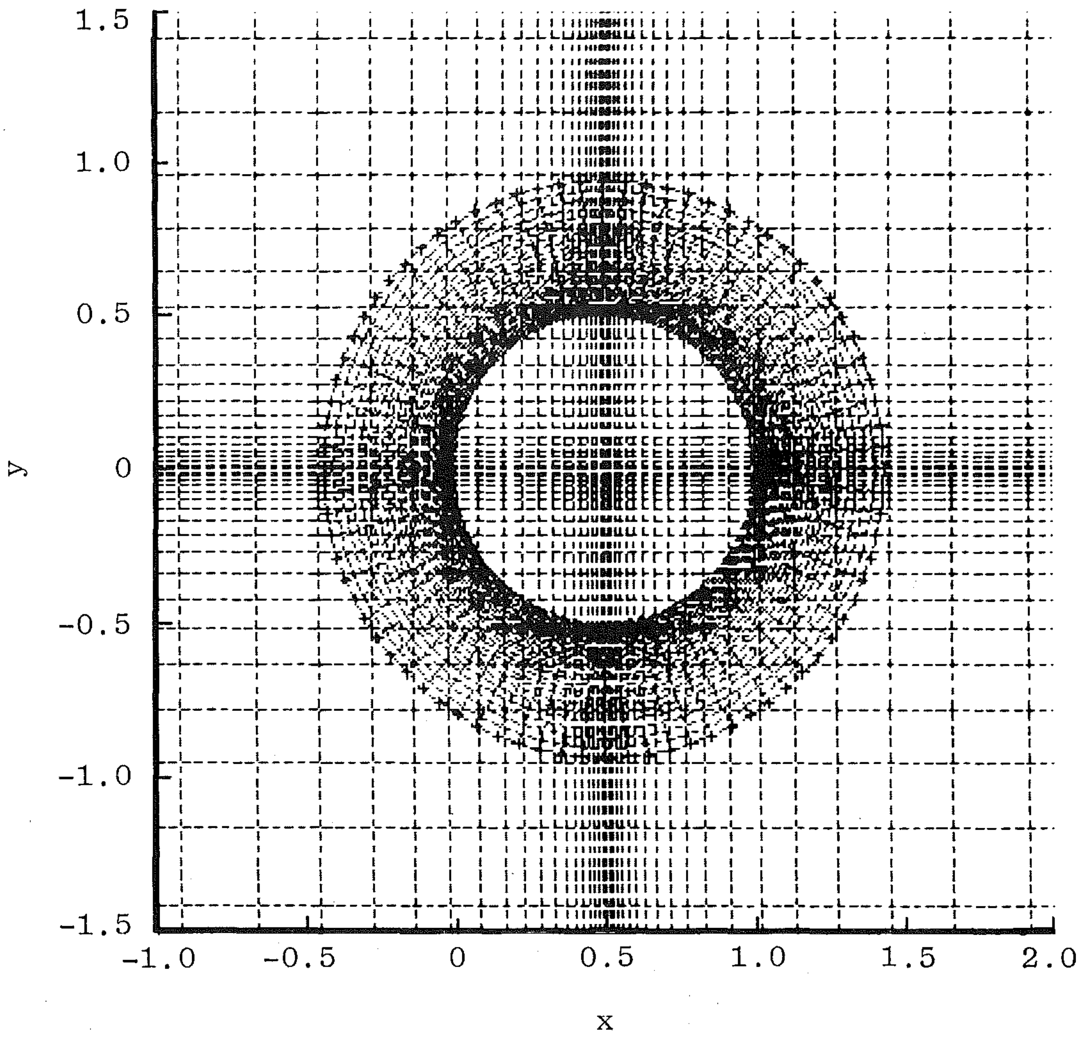


Figure 10. Embedded mesh for circular cylinder in crossflow.

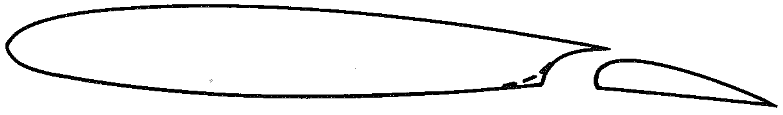


Figure 11. SKF1.1 maneuver flap configuration.

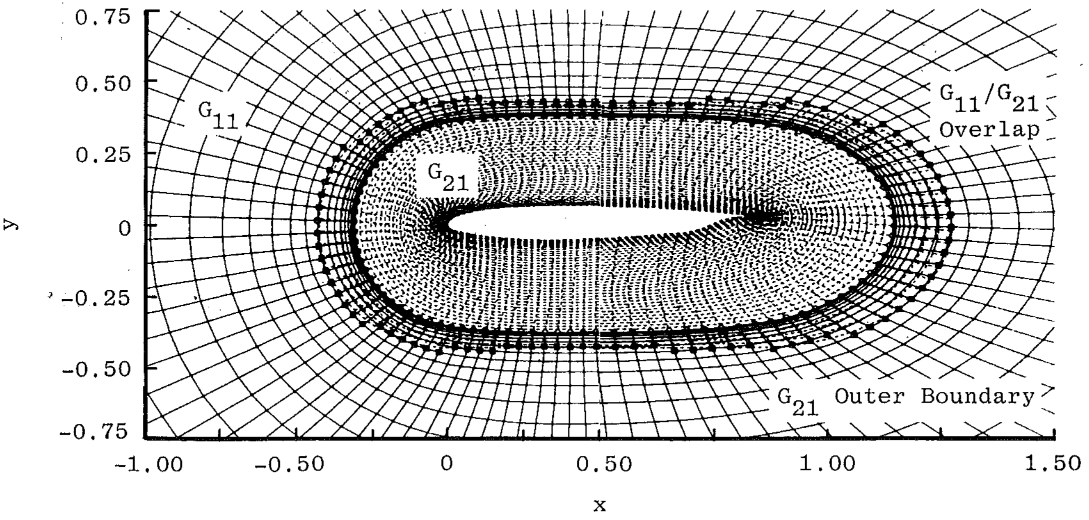


Figure 12. Maneuver flap configuration, grids G_{11} and G_{21}.

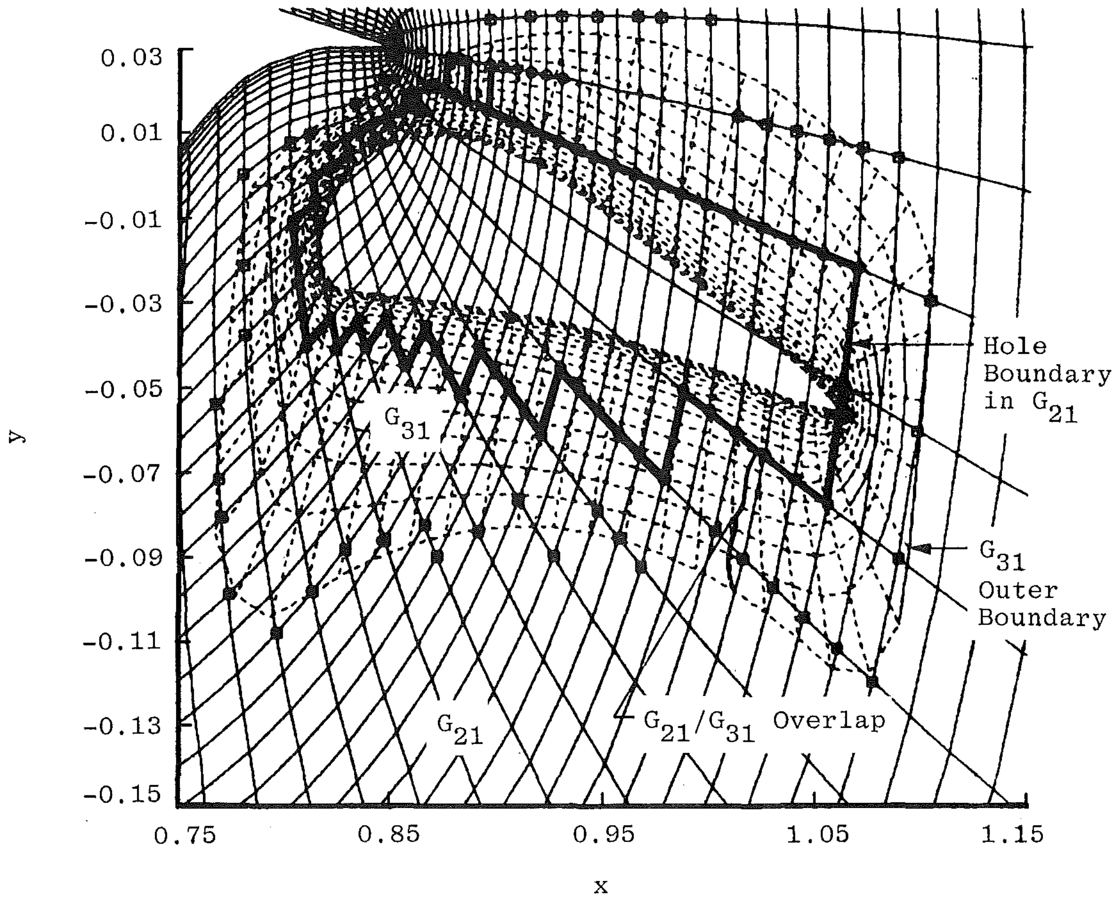


Figure 13. Maneuver flap configuration, grids G_{21} and G_{31}.

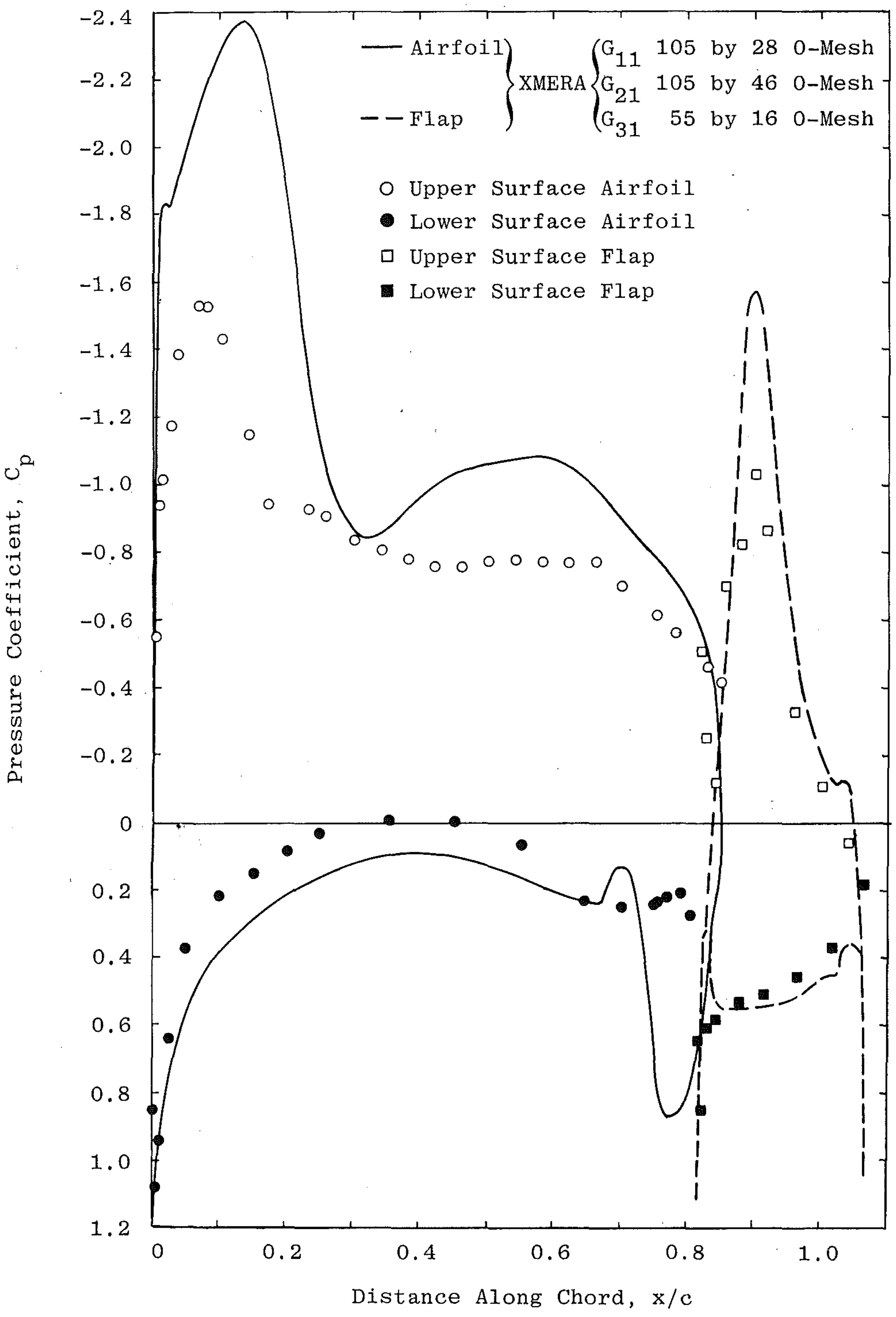


Figure 14. Maneuver flap solution for subcritical flow, $M_\infty = 0.6$, $\alpha = 3$ deg, $\beta = 10$ deg.

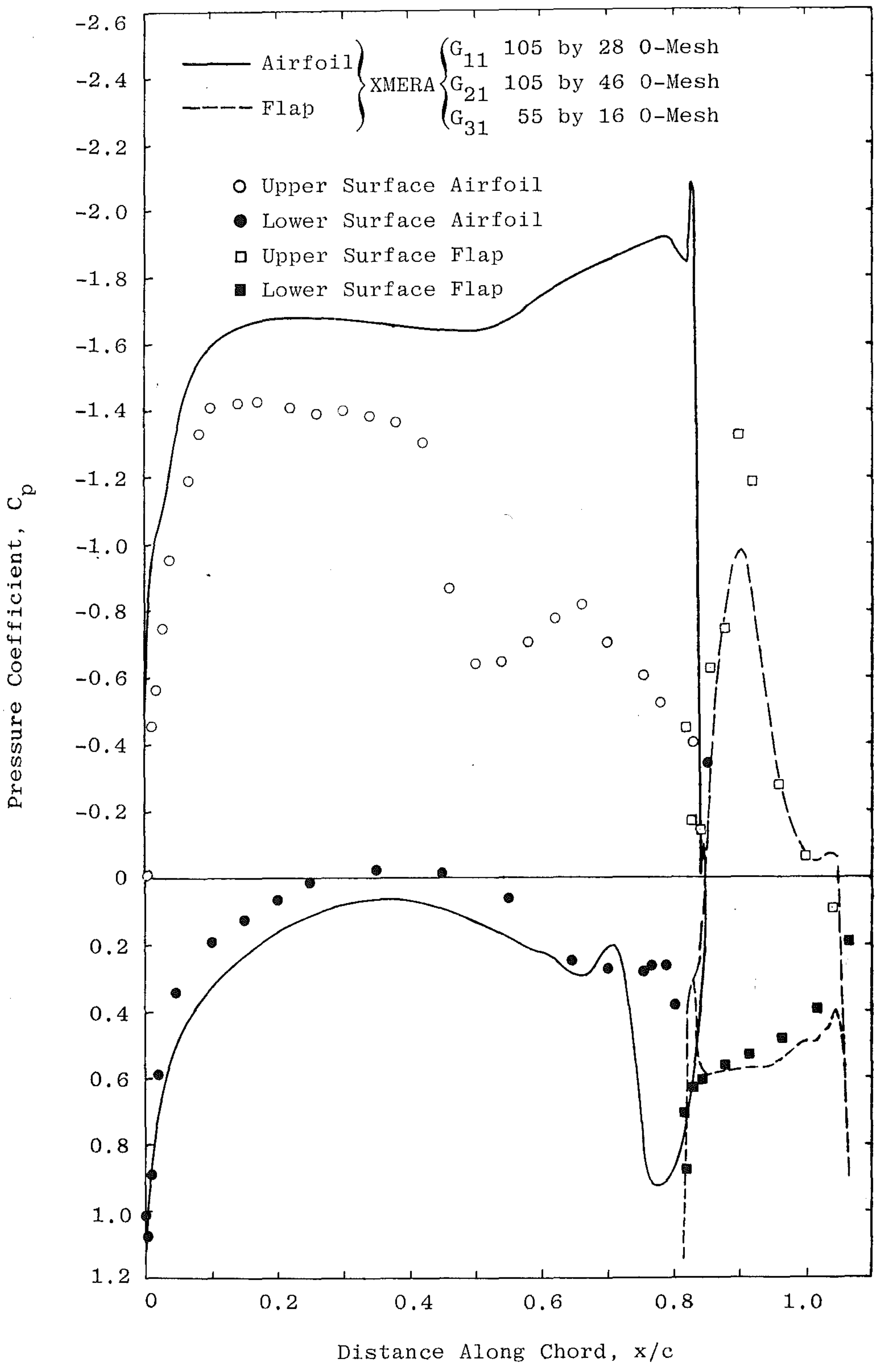


Figure 15. Maneuver flap solution for supercritical flow, $M_\infty = 0.7$, $\alpha = 3$ deg, $\beta = 10$ deg.

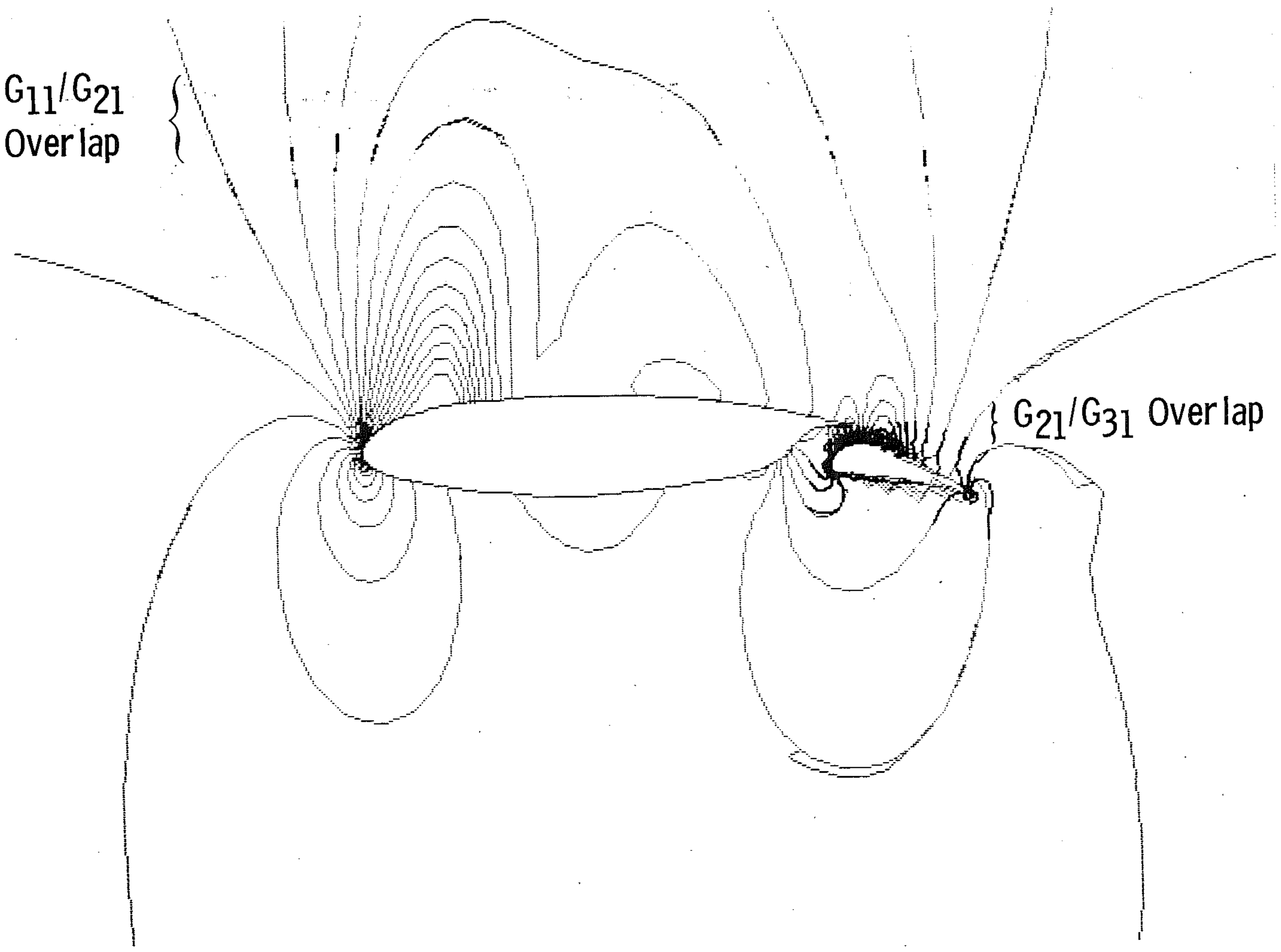


Figure 16. Mach contours for subcritical conditions, $M_\infty = 0.6$, $\alpha = 3$ deg, $\beta = 10$ deg.

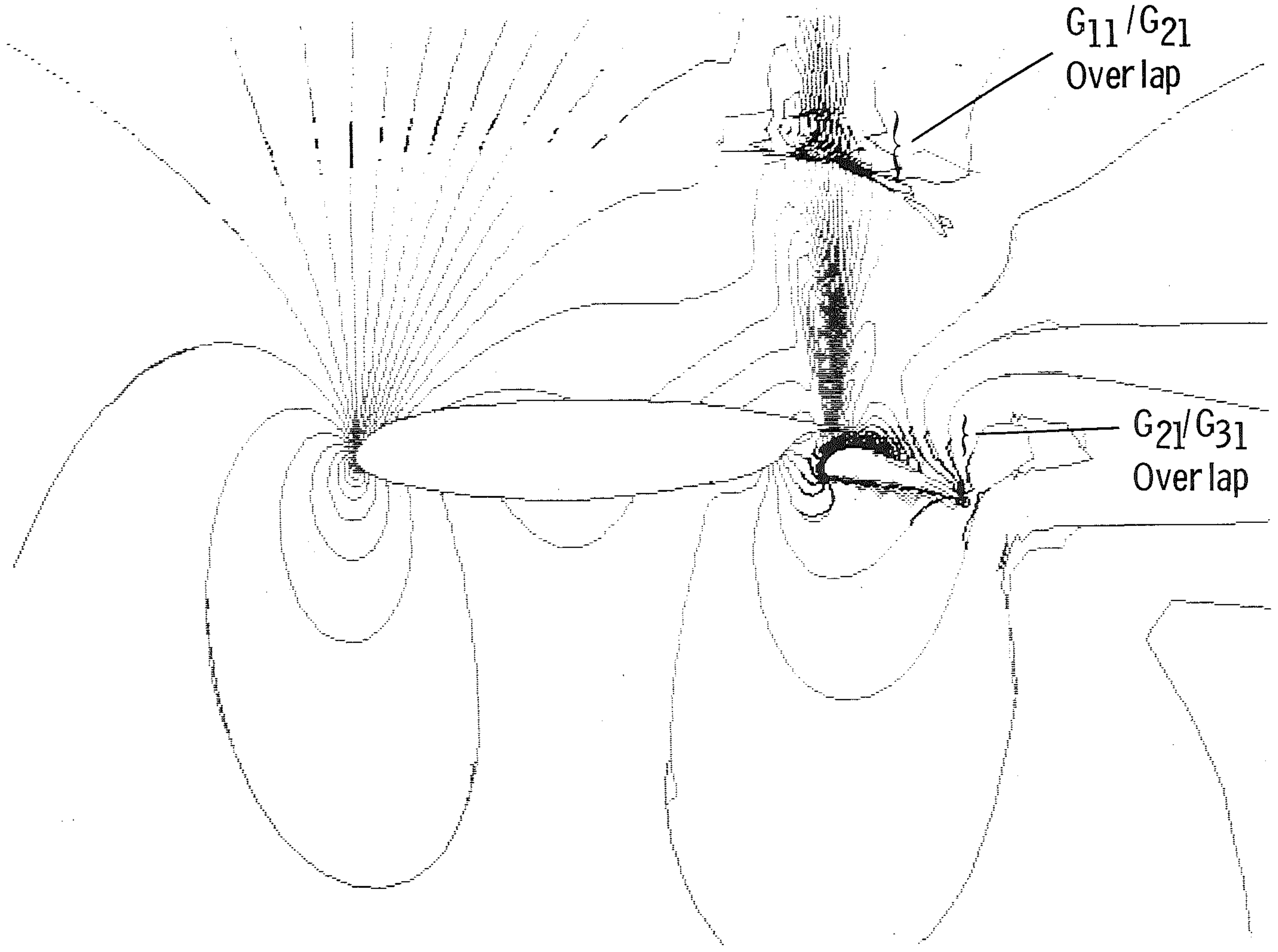


Figure 17. Mach contours for supercritical conditions, $M_\infty = 0.7$, $\alpha = 3$ deg, $\beta = 10$ deg.

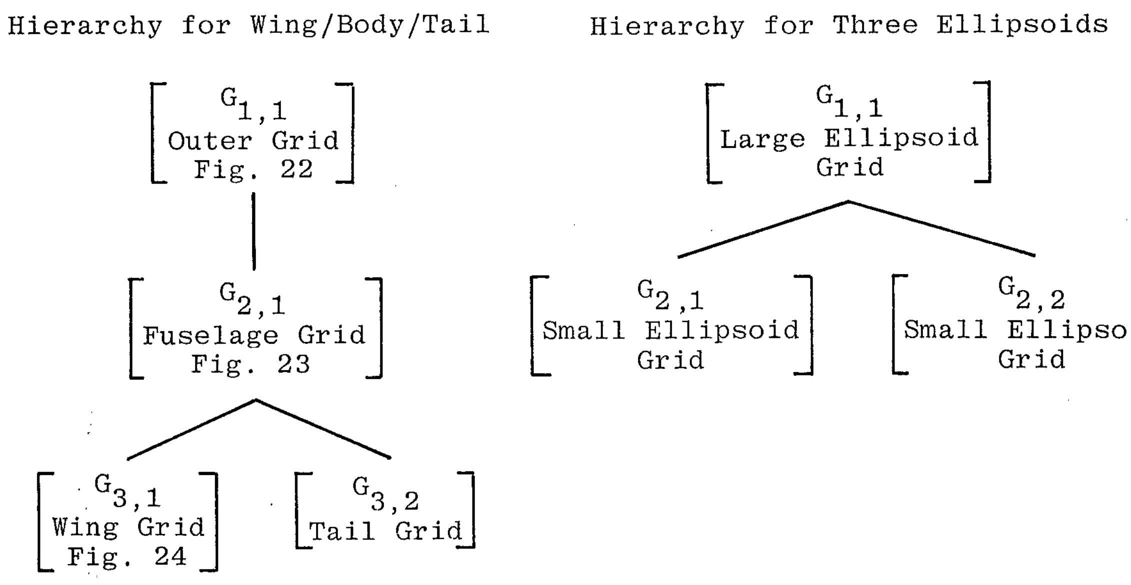

Hierarchy for Wing/Body/Tail
$G_{1,1}$ Outer Grid Fig. 22
$G_{2,1}$ Fuselage Grid Fig. 23
$G_{3,1}$ Wing Grid Fig. 24
$G_{3,2}$ Tail Grid
Hierarchy for Three Ellipsoids
$G_{1,1}$ Large Ellipsoid Grid
$G_{2,1}$ Small Ellipsoid Grid
$G_{2,2}$ Small Ellipsoid Grid

Figure 18. Grid-embedding hierarchies for 3-D applications.

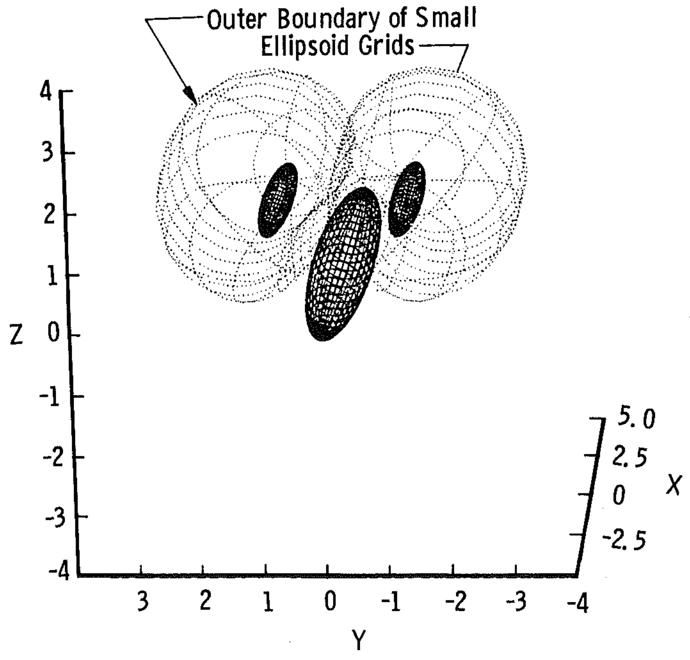


Figure 19. Three-ellipsoid-body configuration and grids.

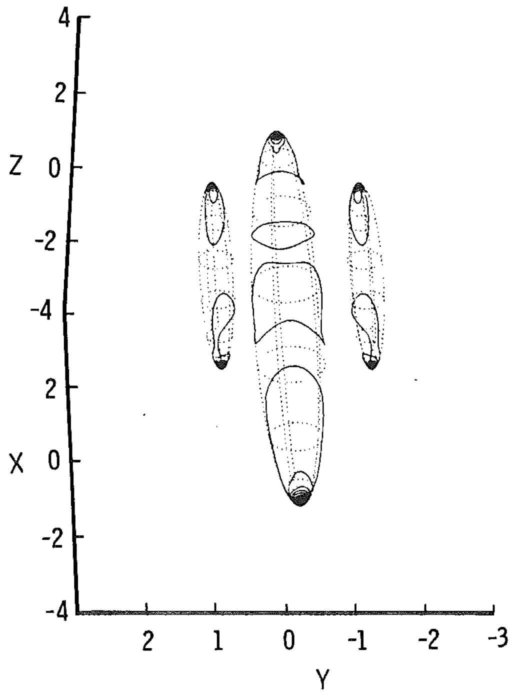


Figure 20. Mach number contours on surfaces of ellipsoids, $M_\infty = 0.80$, $\alpha = -2$ deg.

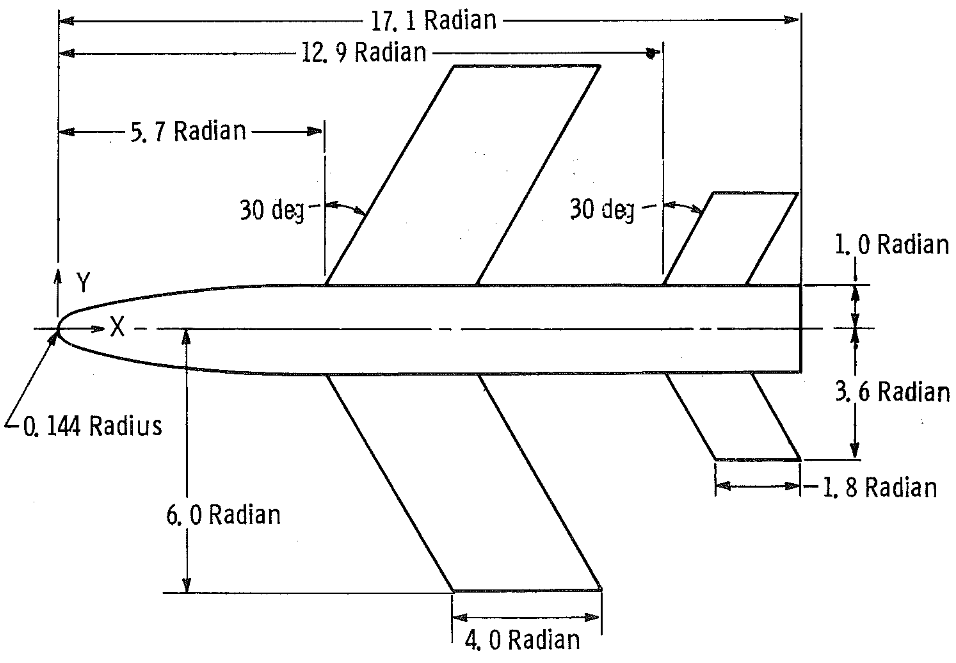


Figure 21. Wing/body/tail configuration.

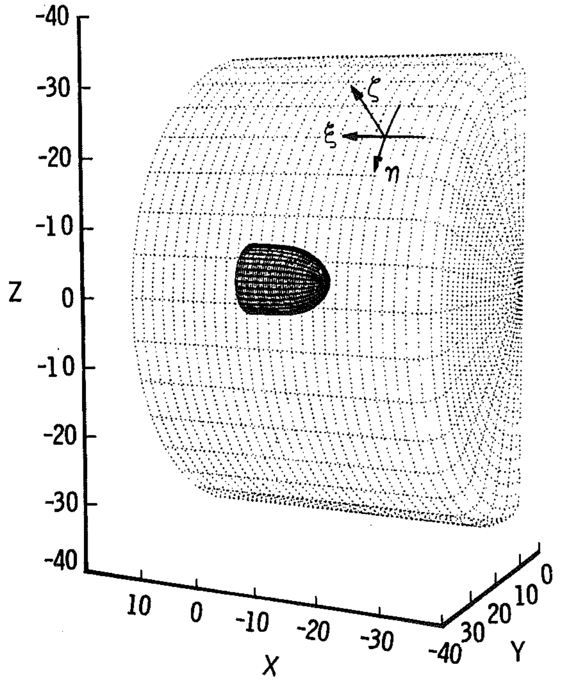


Figure 22. Outer grid for wing/body.

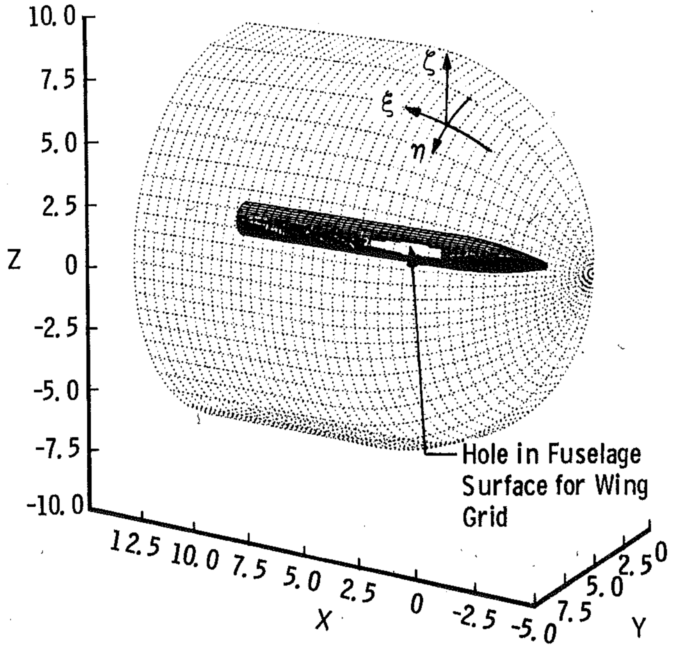


Figure 23. Fuselage grid.

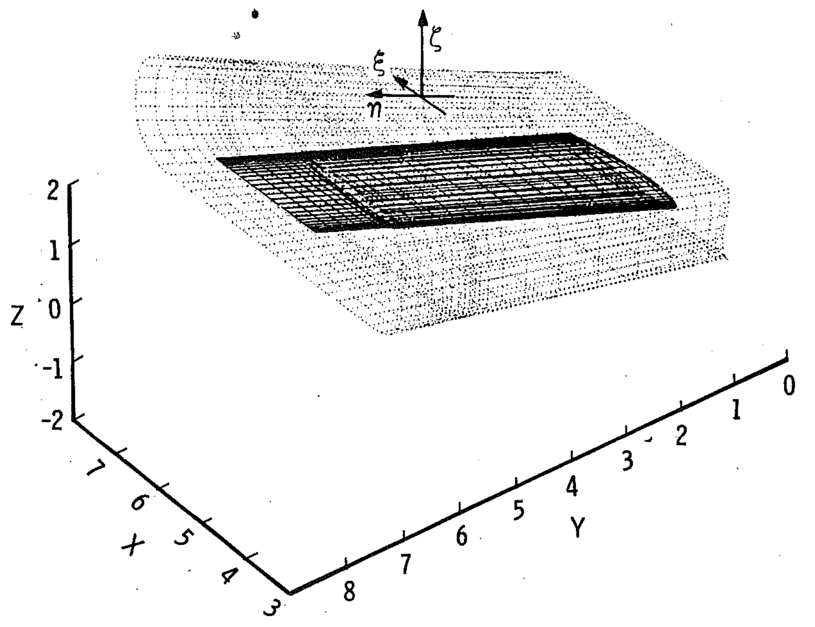


Figure 24. Wing grid.

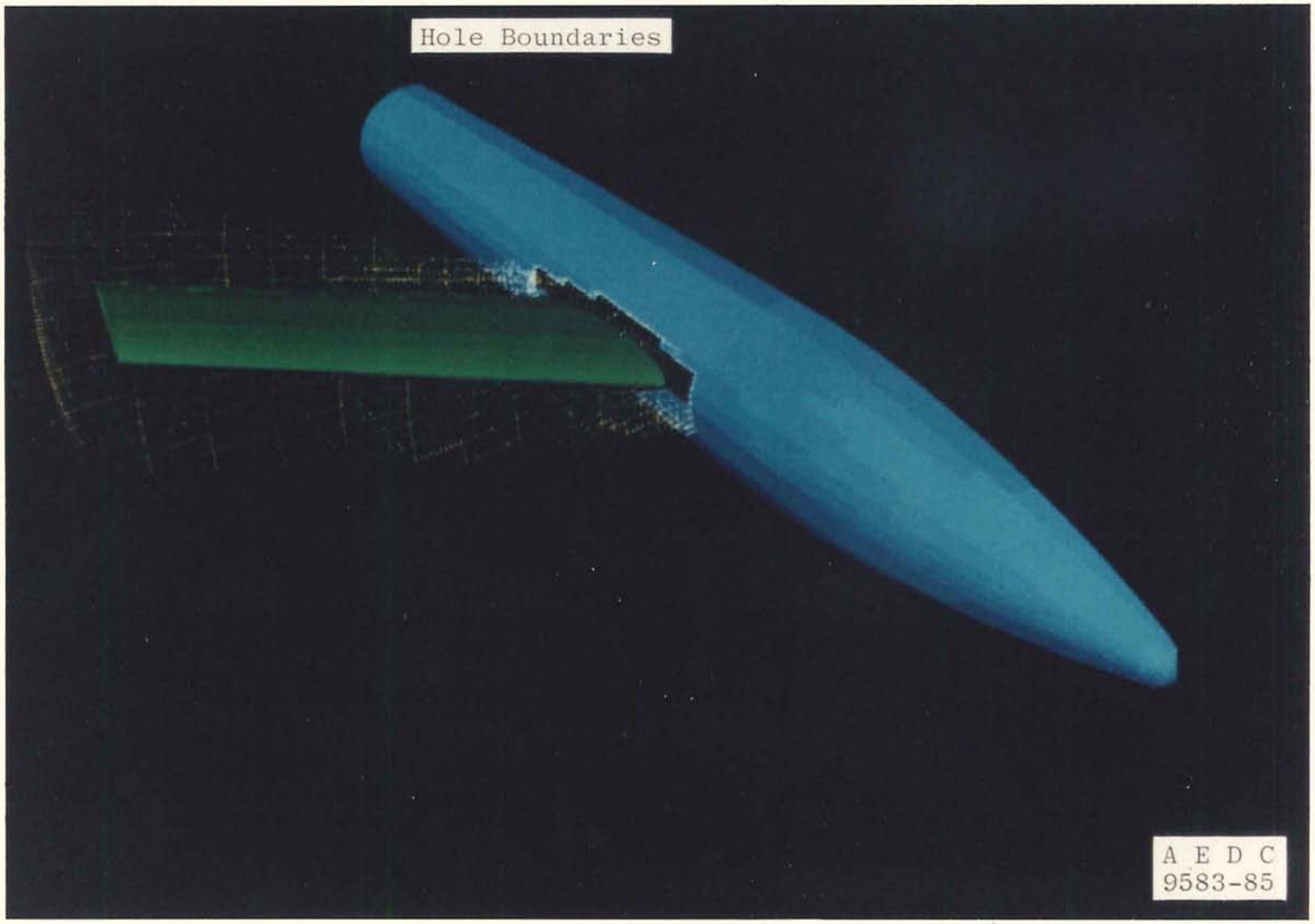


Figure 25. Fuselage mesh hole caused by embedded wing grid.

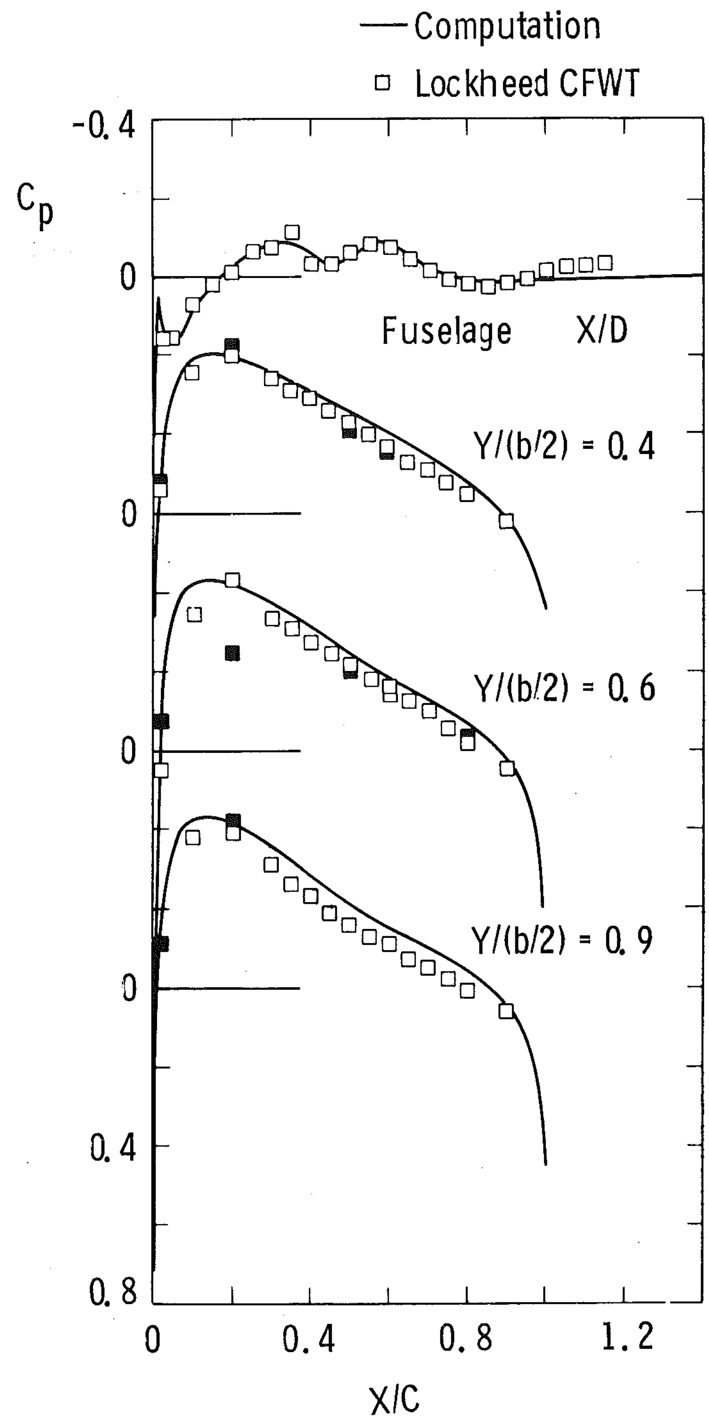


Figure 26. Wing/body solution, M_∞ = 0.60, α = 0 deg (open symbols, upper surface; solid symbols, lower surface).

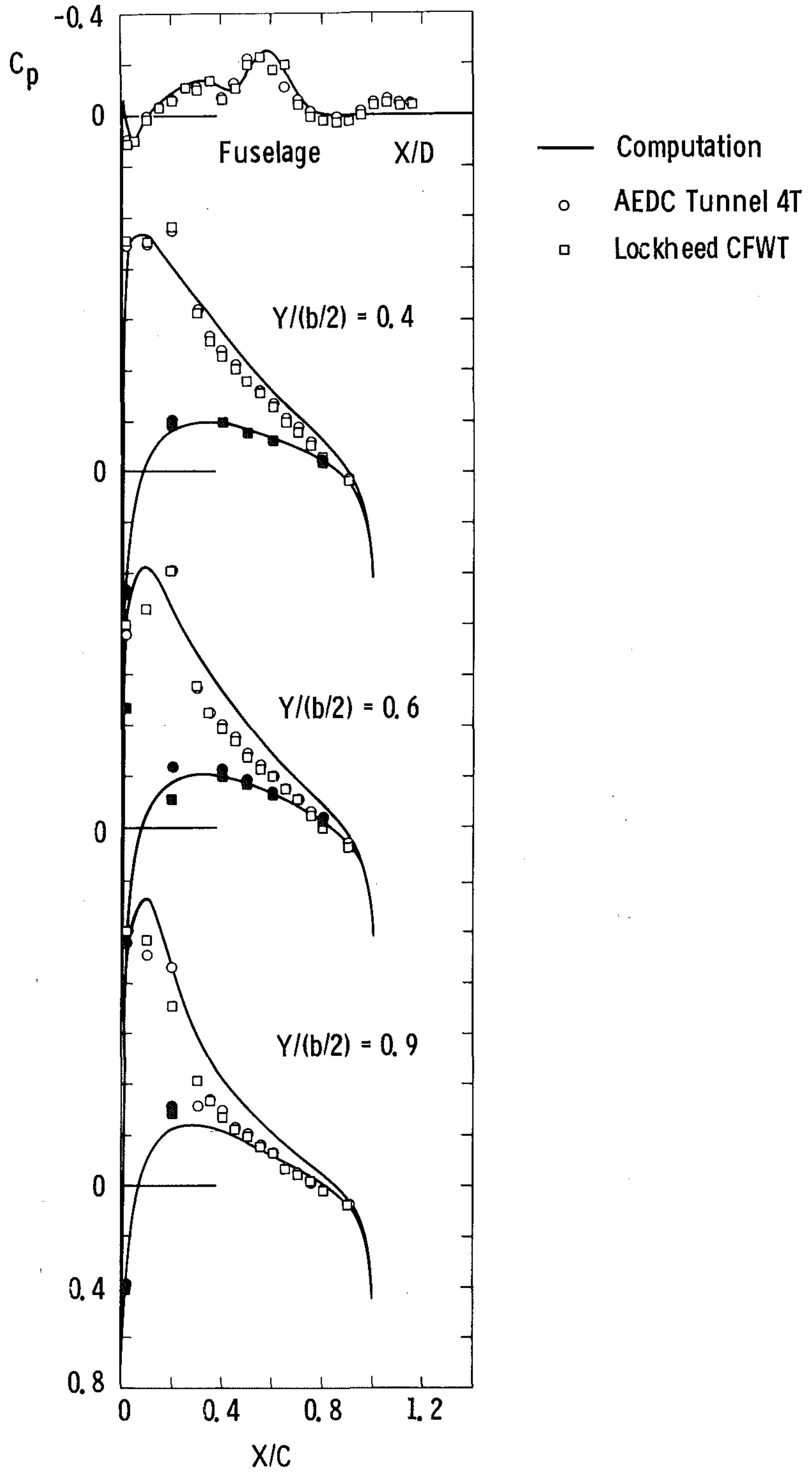


Figure 27. Wing/body solution, M_∞ = 0.75, α = 4 deg (open symbols, upper surface; solid symbols, lower surface).

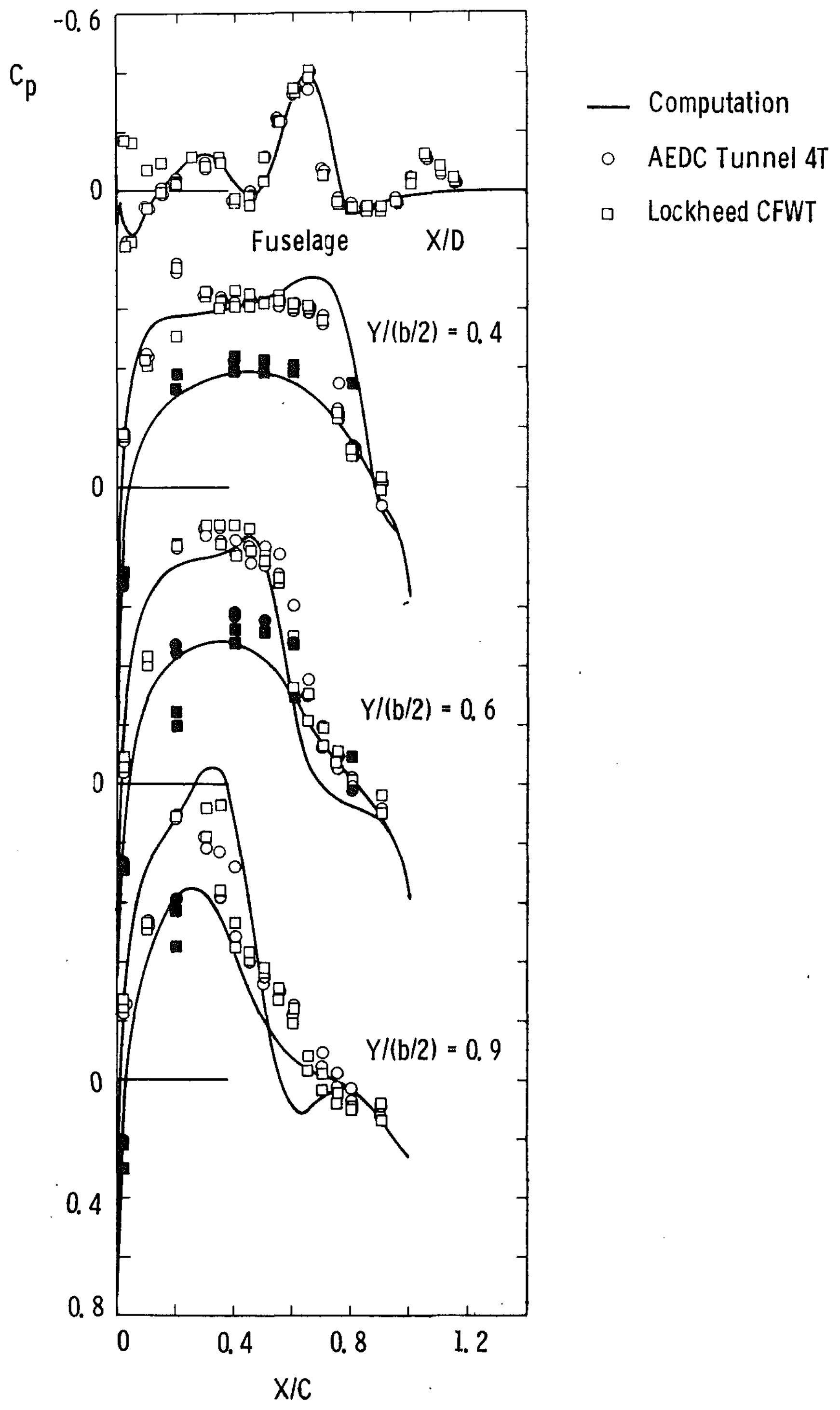
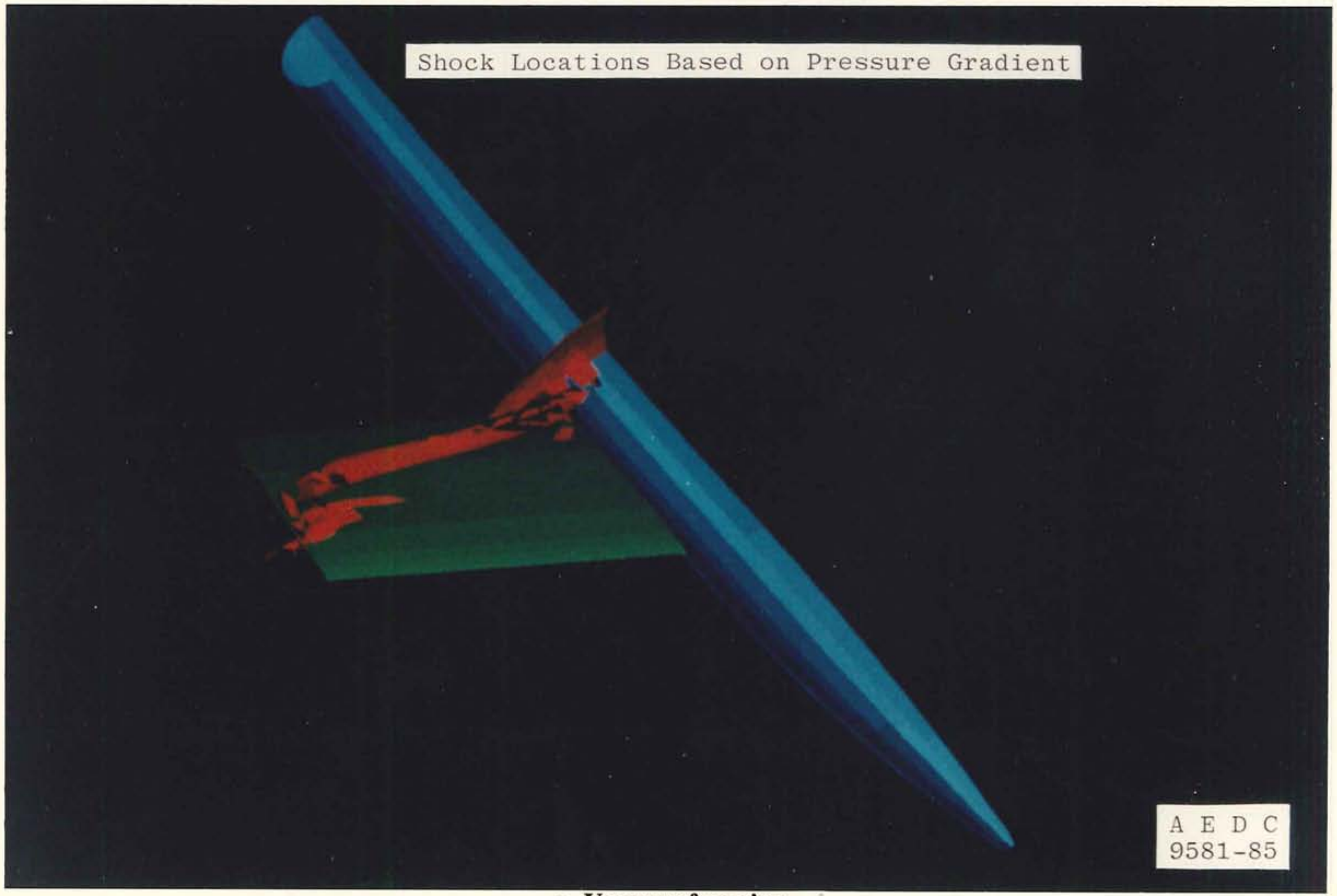
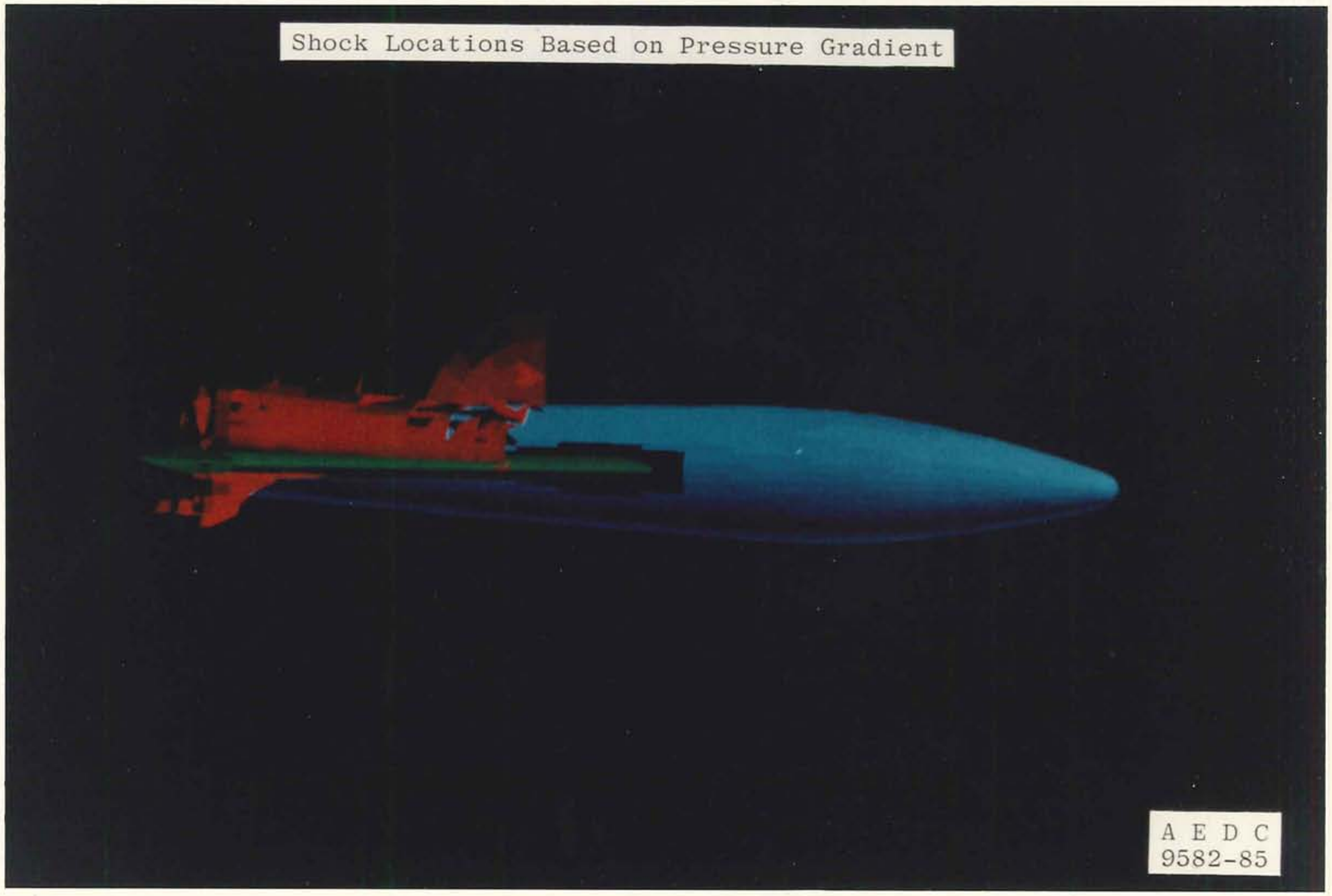


Figure 28. Wing/body solution, $M_\infty = 0.90$, $\alpha = 2$ deg (open symbols, upper surface; solid symbols, lower surface).



a. Upper-surface view

Figure 29. Wing/body shock surface, $M_\infty = 0.90$, $\alpha = 2$ deg as determined by PLOT3D.



b. View from planform plane
Figure 29. Concluded.

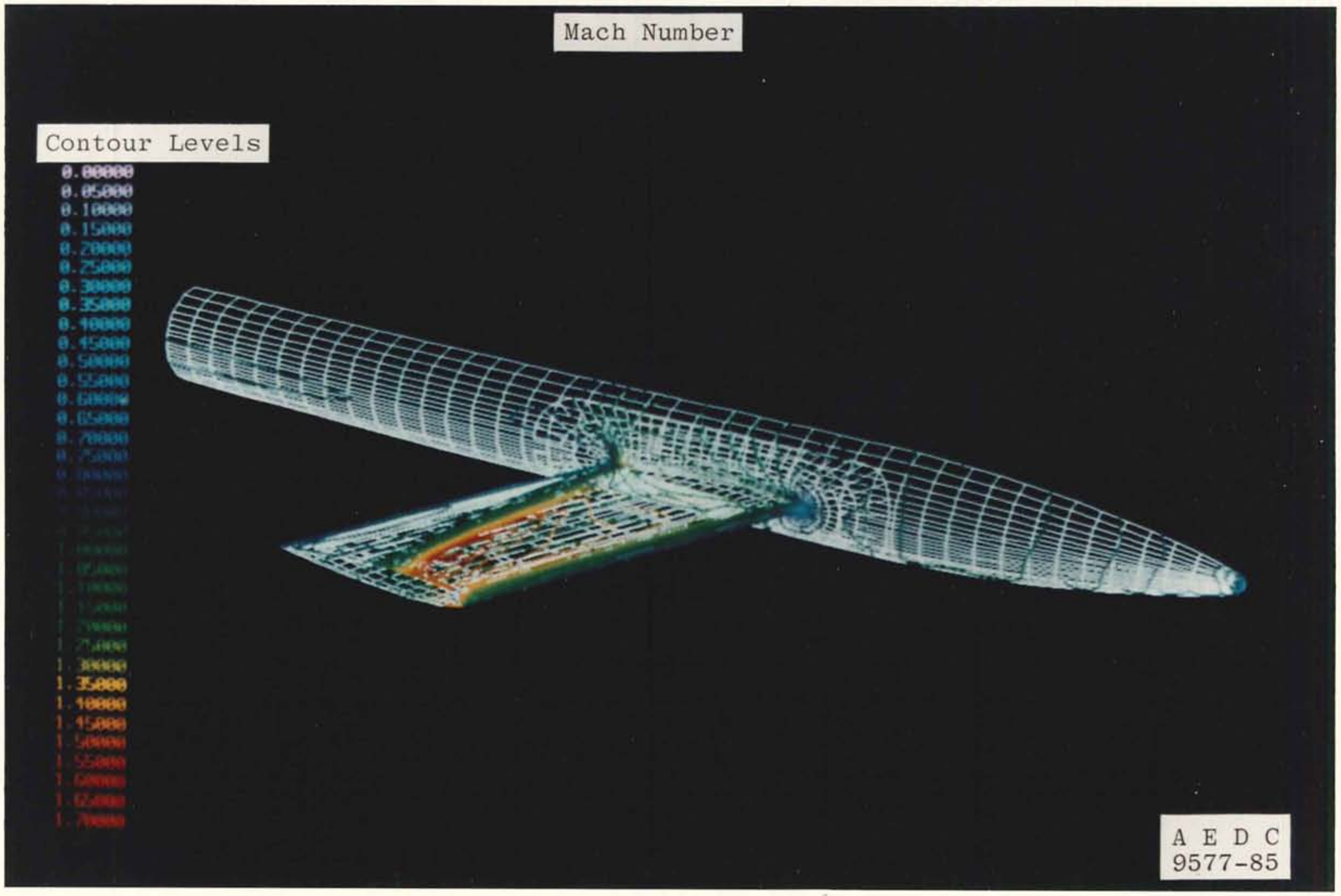


Figure 30. Mach number contours on wing/body surface, $M_\infty = 0.90$, $\alpha = 2$ deg.

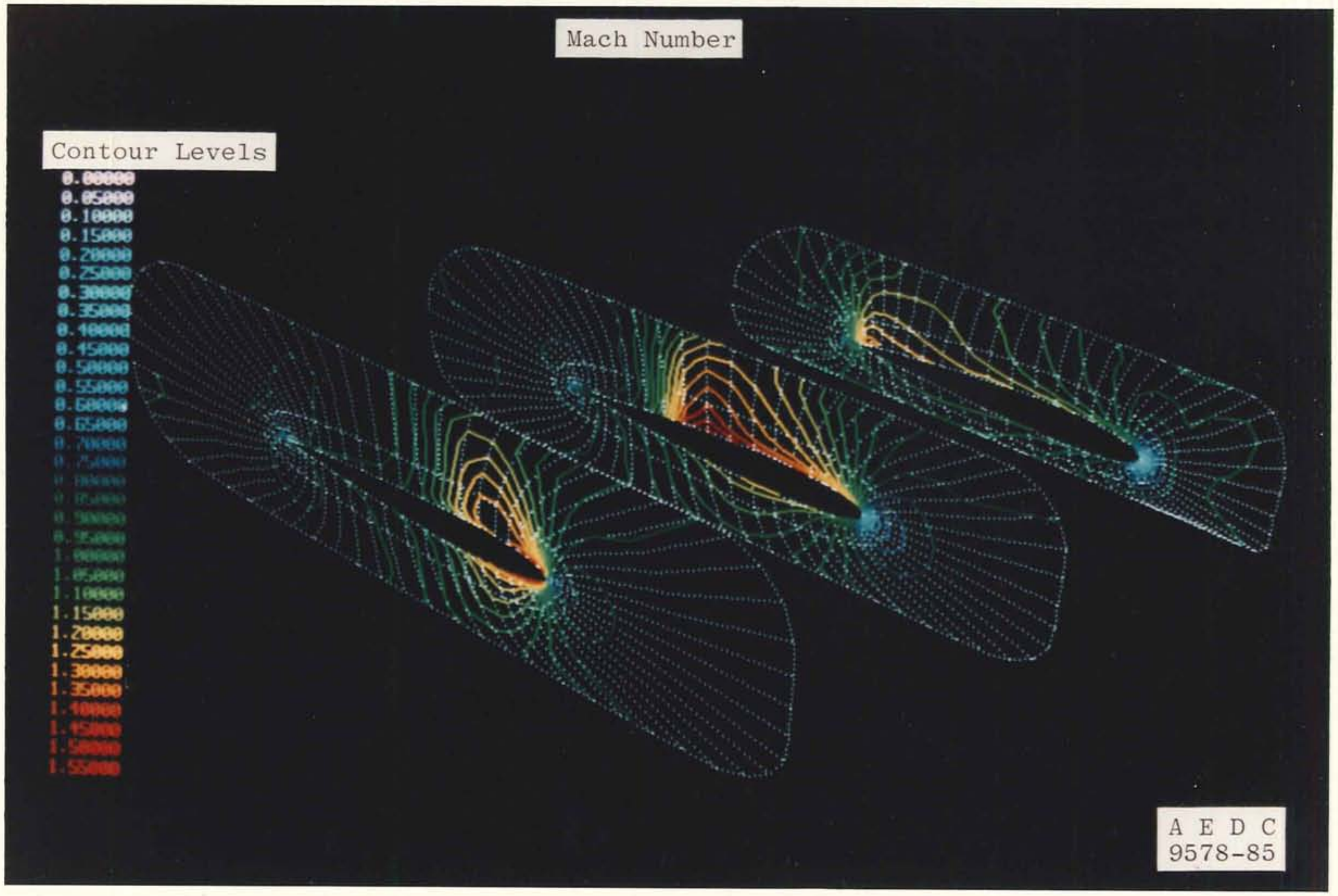


Figure 31. Mach number contours in three η = constant planes, M_∞ = 0.90, α = 2 deg.

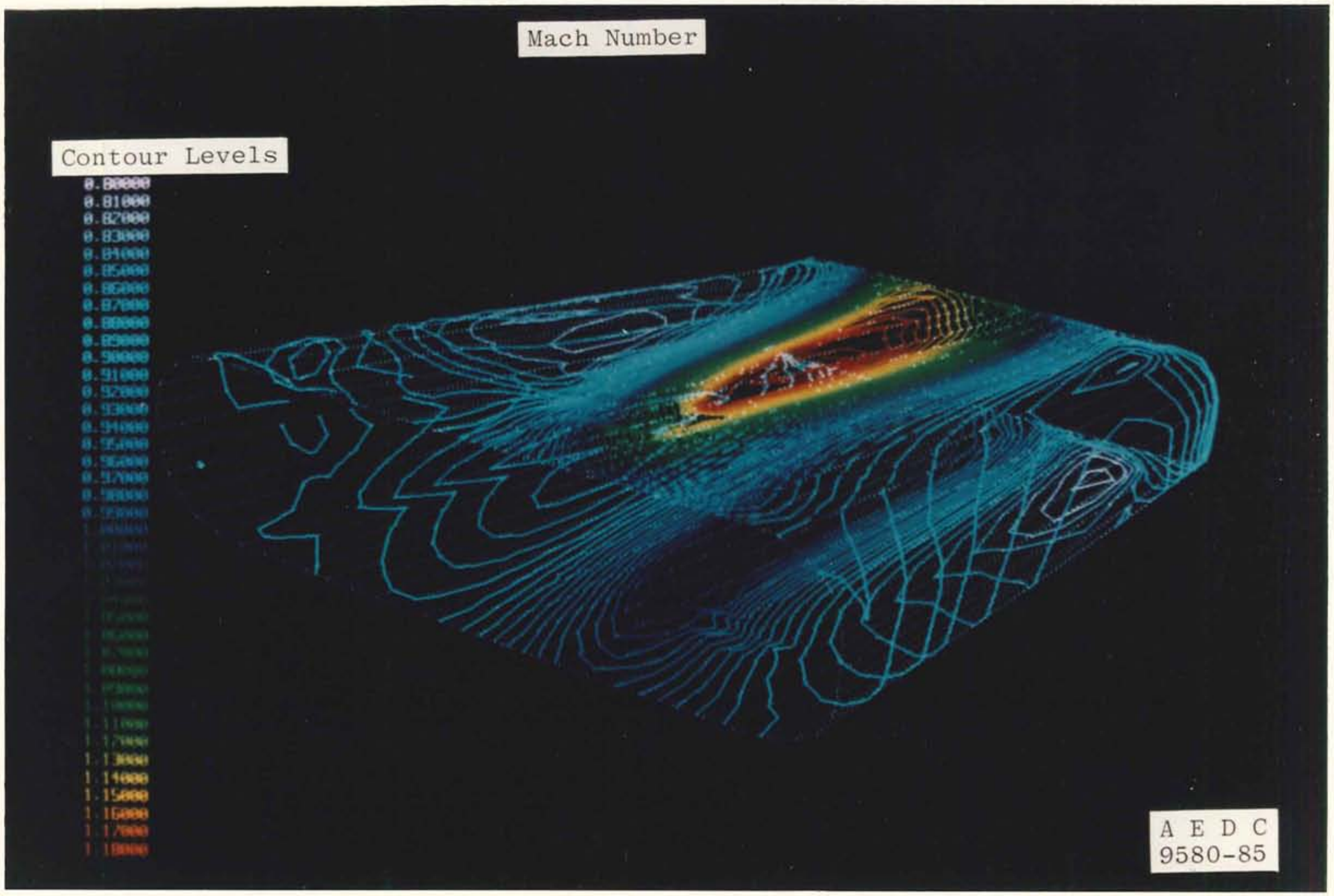


Figure 32. Mach number contours on outer boundary of wing grid, $M_{\infty} = 0.90$, $\alpha = 2$ deg.

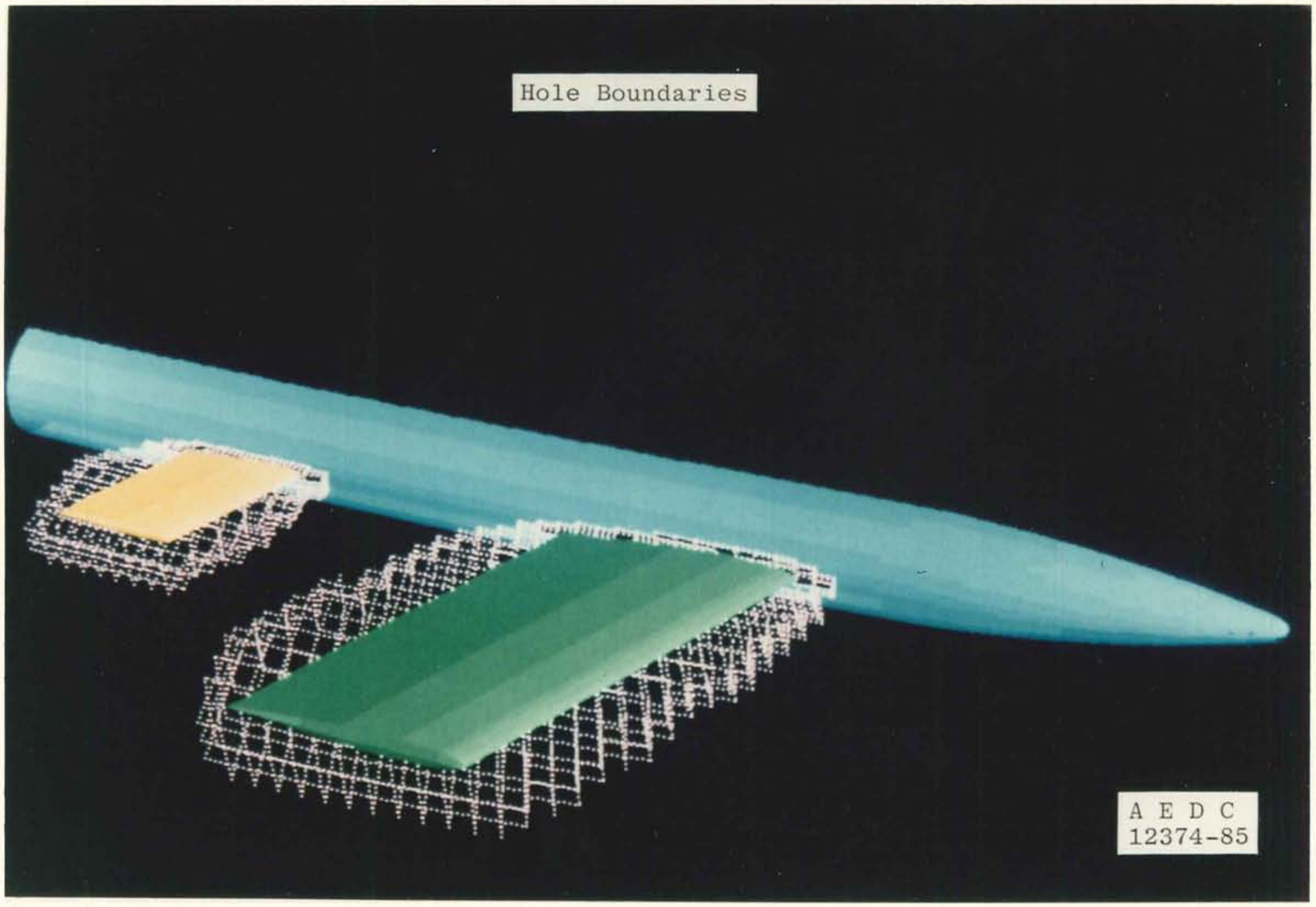


Figure 33. Fuselage mesh holes caused by embedded wing and tail grids.

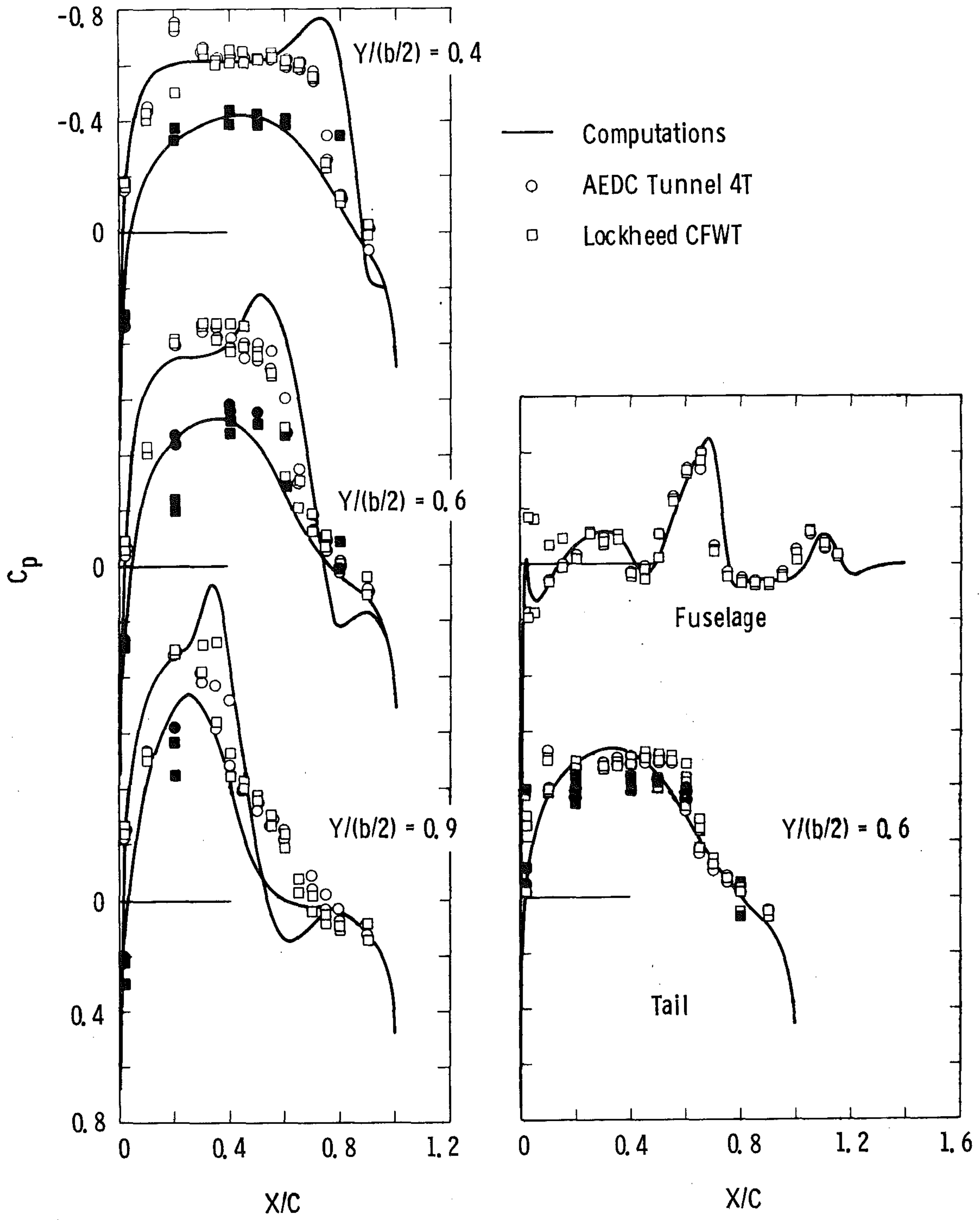


Figure 34. Wing/body/tail solution, M_∞ = 0.90, α = 2 deg (open symbols, upper surface; solid symbols, lower surface).

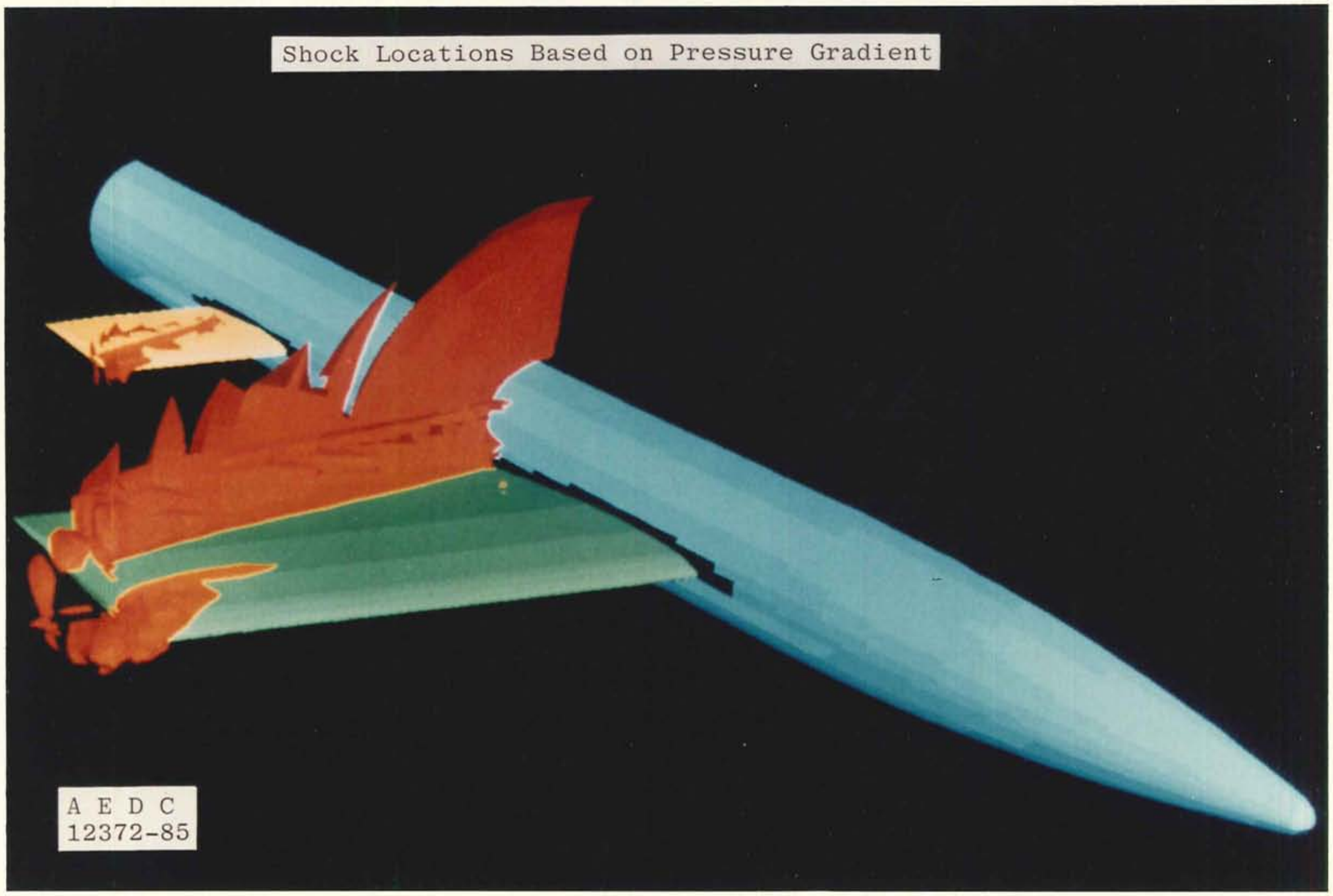


Figure 35. Wing/body/tail shock surface, $M_\infty = 0.90$, $\alpha = 2$ deg.

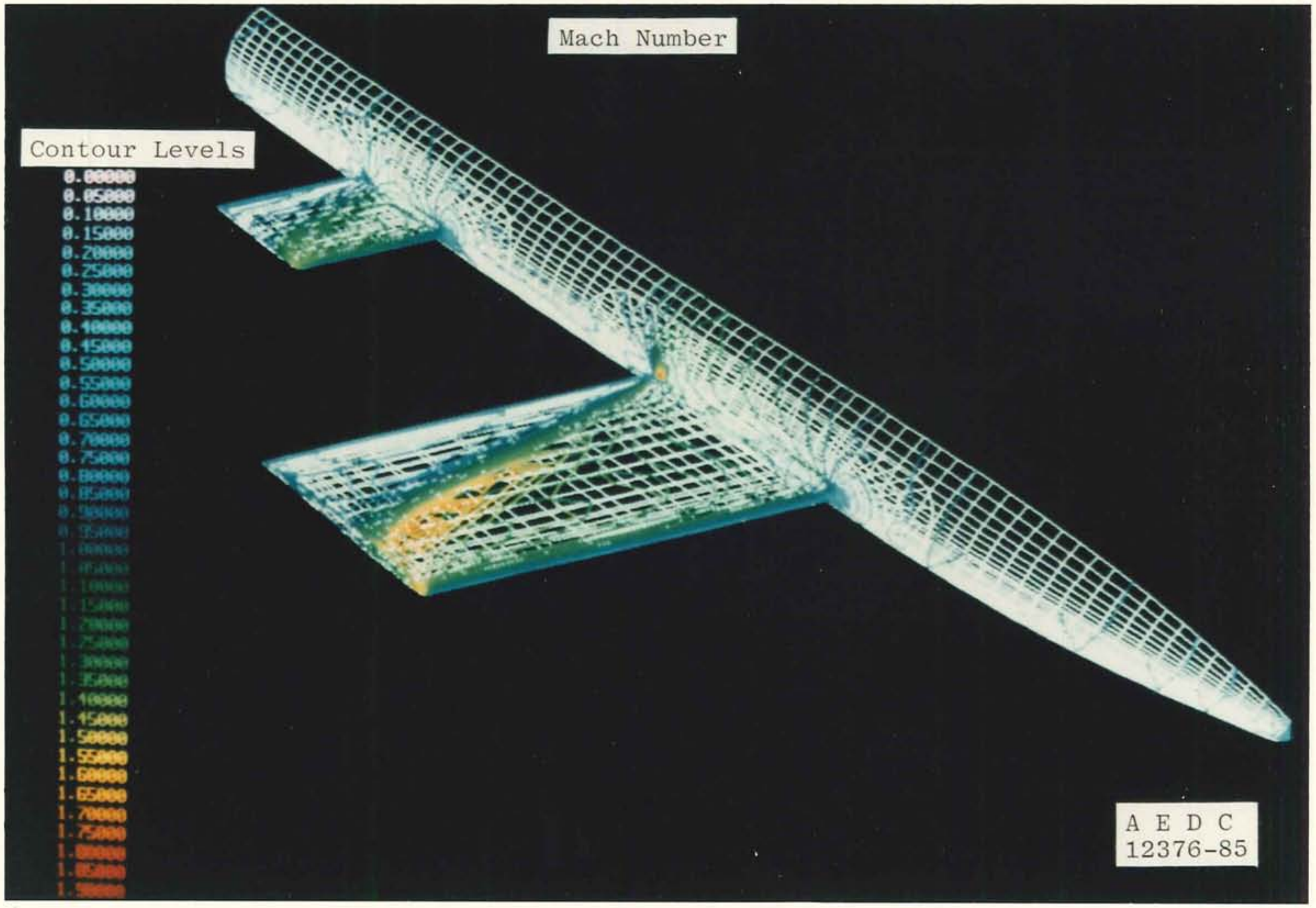


Figure 36. Mach number contours on wing/body/tail surface, $M_\infty = 0.90$, $\alpha = 2$ deg.

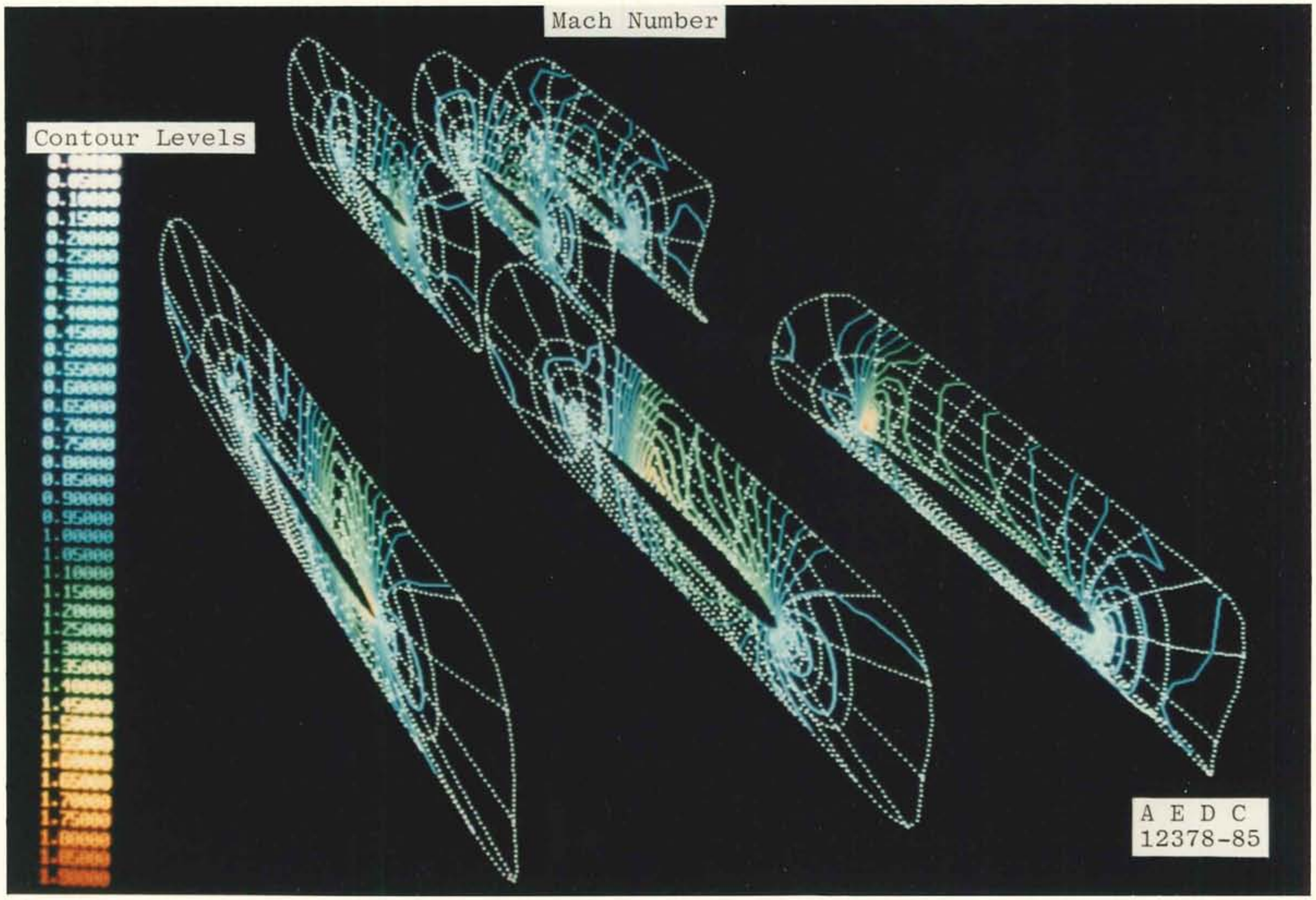


Figure 37. Mach number contours in three η = constant surfaces in wing and tail grids, M_∞ = 0.90, α = 2 deg.

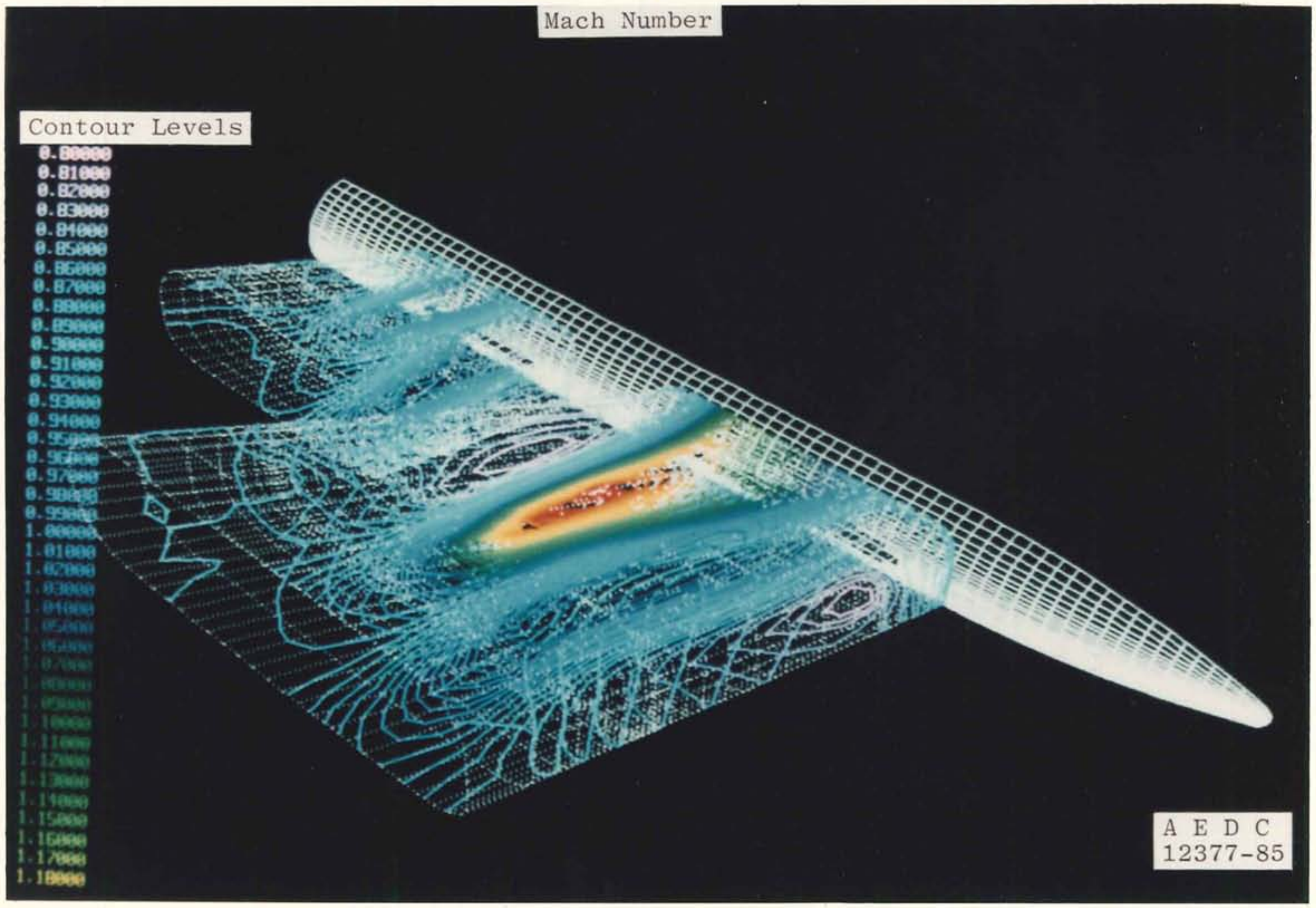


Figure 38. Mach number contours on outer boundary of wing and tail grids, $M_\infty = 0.90$, $\alpha = 2$ deg.

APPENDIX A
STRUCTURE CHART FOR PEGSUS

The structure charts (Figs. A-1, A-2, and A-3) for the PEGSUS Code clarify the conceptual components of the program and the relationships among them. The conceptual elements are arranged in a hierarchy with the most general components on the highest levels and the most specialized on the lowest. Whenever a specific element is accomplished in a single subroutine, it is identified on the structure chart by SXX, where XX is the number of an entry in Table A-1 which identifies the subroutine by name. Thus, the charts also illustrate the calling sequence of subroutines. Note that the charts may identify the same conceptual element in more than one place. This repetition occurs for purposes of clarity. Similarly, namelist data inputs are indicted as NLXX and are identified in Table A-2. For details of the functions performed in each subroutine, see Appendix D; for details of the input data, see Appendix F.

Table A-1 Subroutine Names for PEGSUS Structure Chart

Number	Subroutine Name
S1	INITIA
S2	COMPOS
S3	OUTPUT
S4	WCOORD
S5	CHKPLT
S6	HOLE
S7	OUTER
S8	RGRID
S9	TRANS
S10	CHKOUT
S11	CHKSTN
S12	CINDEX
S13	WIBLNK
S14	HDATA
S15	INTDAT
S16	HINTPT
S17	PLTHOL
S18	INITHB
S19	FRNGE
S20	PLTIBL

Table A-1 Concluded

Number	Subroutine Name
S21	HLOCAT
S22	SETPTR
S23	QUAD
S24	NEARPT
S25	NORMAL
S26	ODATA
S27	OLOCAT
S28	OBOUND
S29	PLTOI

Table A-2 Namelist Names for PEGSUS Structure Chart

Number	Subroutine Name
NL1	HIERCY
NL2	SEARCH
NL3	CKPLOT
NL4	GRDPRM
NL5	HBOUN
NL6	OBOUN

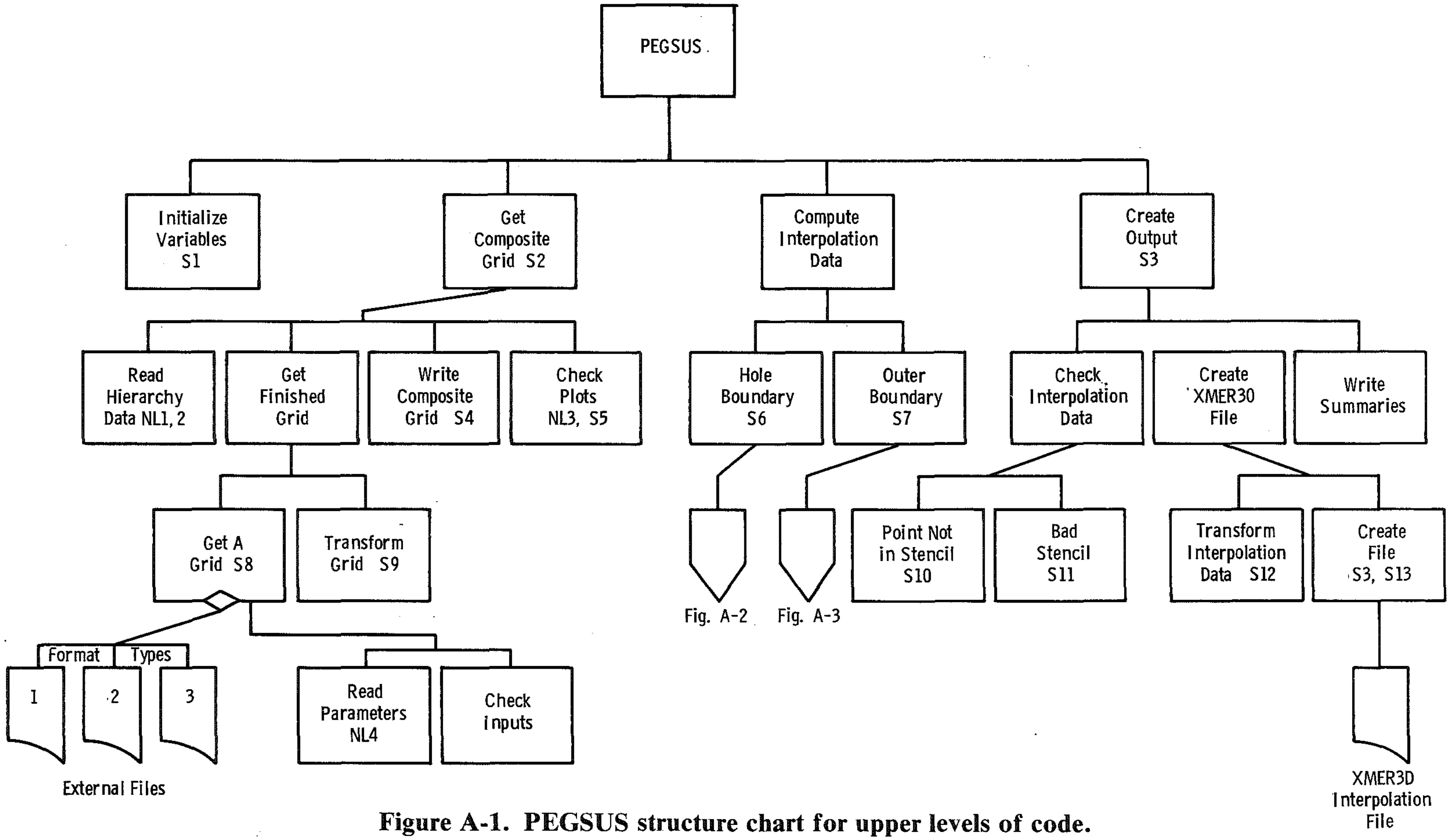


Figure A-1. PEGSUS structure chart for upper levels of code.

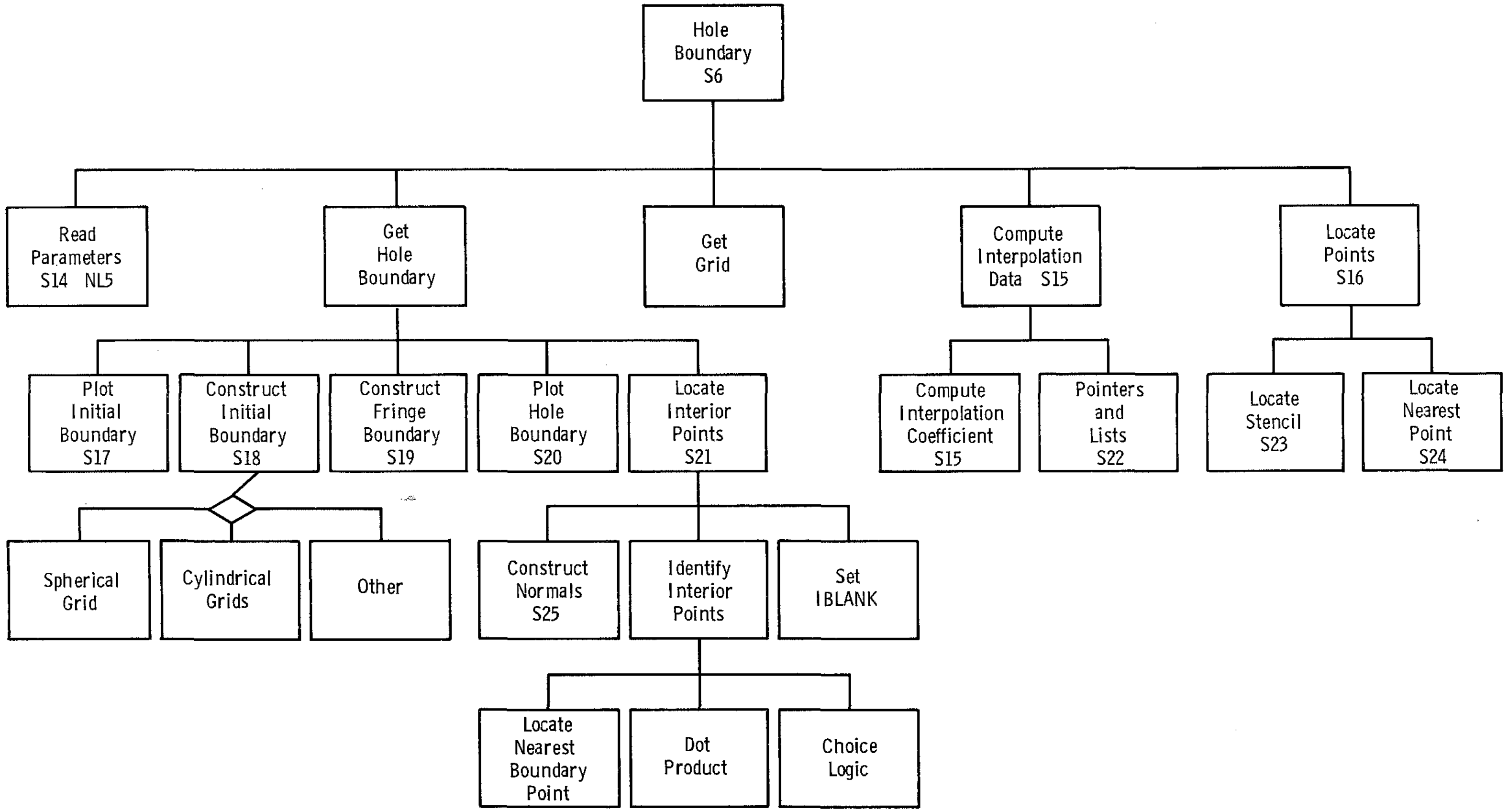


Figure A-2. PEGSUS structure chart detailing hole-construction logic.

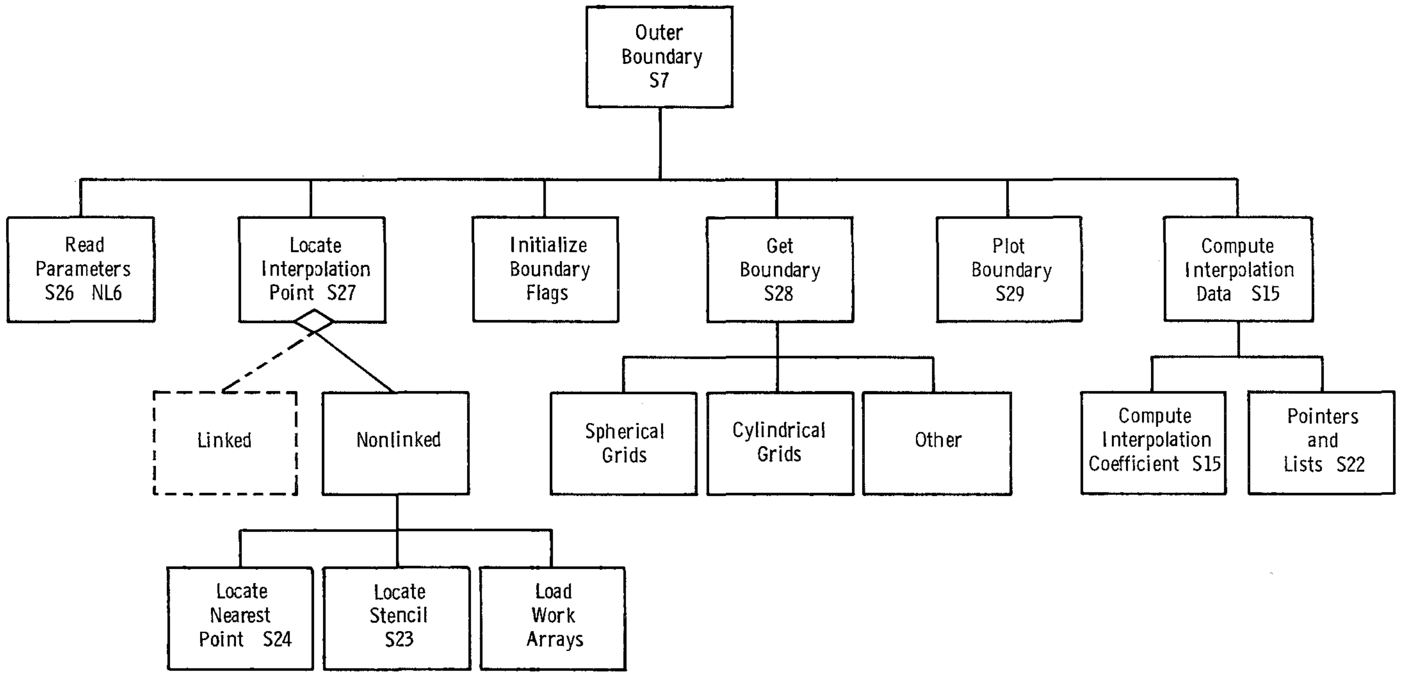


Figure A-3. PEGSUS structure chart detailing outer-boundary logic.

APPENDIX B
TRILINEAR INTERPOLATION

Trilinear interpolation can only be used on cubes. Unfortunately, the typical cell resulting from grid generation in curvilinear coordinates is a warped hexahedron. Therefore, each cell containing a point at which a function value is to be interpolated must first be transformed to a cube (Fig. B-1). This is most easily accomplished by applying the same isoparametric form to the coordinates of the hexahedron as is used for the interpolation. This is

$$f = a_1 + a_2 \bar{\xi} + a_3 \bar{\eta} + a_4 \bar{\zeta} + a_5 \bar{\xi} \bar{\eta} + a_6 \bar{\xi} \bar{\zeta} + a_7 \bar{\eta} \bar{\zeta} + a_8 \bar{\xi} \bar{\eta} \bar{\zeta} \qquad \text{(B-1)}$$

where the a_i, $i = 1, \ldots 8$ are coefficients depending on the values of f at the vertices of the cube, and $(\bar{\xi}, \bar{\eta}, \bar{\zeta})$ are coordinates of the interpolated point, P, relative to a vertex of the cube. For convenience, we map to the unit cube (See Fig. B-1), so

$$0 \leq \xi, \eta, \zeta \leq 1 \qquad \text{(B-2)}$$

The coefficients a_i can easily be obtained from the values of f at the vertices of the cube. For example, at $(\bar{\xi}, \bar{\eta}, \bar{\zeta}) = (0, 0, 0)$, $f_1 = a_1$, where f_1 is the value of f at vertex 1 (See Fig. B-1). Repetition of this procedure leads to the system

$$f_1 = a_1$$

$$f_2 = a_1 + a_2$$

$$f_3 = a_1 + a_2 + a_3 + a_5$$

$$f_4 = a_1 + a_3$$

$$f_5 = a_1 + a_4$$

$$f_6 = a_1 + a_2 + a_4 + a_6$$

$$f_7 = a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8$$

$$f_8 = a_1 + a_3 + a_4 + a_7 \qquad \text{(B-3)}$$

Solution for the a_i in terms of the f_i yields

$$a_1 = f_1$$

$$a_2 = -f_1 + f_2$$

$$a_3 = -f_1 + f_4$$

$$a_4 = -f_1 + f_5$$

$$a_5 = f_1 - f_2 + f_3 - f_4$$

$$a_6 = f_1 - f_2 - f_5 + f_6$$

$$a_7 = f_1 - f_4 - f_5 + f_8$$

$$a_8 = -f_1 + f_2 - f_3 + f_4 + f_5 - f_6 + f_1 - f_8 \qquad \text{(B-4)}$$

We now identify the origin of the cube in interpolation space with the coordinates in physical space as

$$(0, 0, 0) = (X, Y, Z)_{j,\,k,\,l}$$

Hence,

$$f_1 = f_{j,k,l}$$

The subscripts (j, k, l) corresponding to the vertices become

$$f_1 = f_{j,k,l}$$

$$f_2 = f_{j+1,k,l}$$

$$f_3 = f_{j+1,k+1,l}$$

$$f_4 = f_{j,k+1,l}$$

$$f_5 = f_{j,k,l+1}$$

$$f_6 = f_{j+1,k,l+1}$$

$$f_7 = f_{j+1,k+1,l+1}$$

$$f_8 = f_{j,k+1,l+1} \tag{B-5}$$

Thus, the interpolation stencil is specified by specifying (j,k,l) which simplifies the storage requirements.

The mapping of the warped hexahedron to a cube using the same isoparametric mapping as for f defines the transform from $(\bar{\xi}, \bar{\eta}, \bar{\zeta})$ to (X,Y,Z). Thus,

$$X = a_1 + a_2 \bar{\zeta} + a_3 \bar{\eta} + a_4 \bar{\zeta} + a_5 \bar{\xi} \bar{\eta} + a_6 \bar{\xi} \bar{\zeta} + a_7 \bar{\eta} \bar{\zeta} + a_8 \bar{\xi} \bar{\eta} \bar{\zeta}$$

$$Y = b_1 + b_2 \bar{\zeta} + b_3 \bar{\eta} + b_4 \bar{\zeta} + b_5 \bar{\xi} \bar{\eta} + b_6 \bar{\xi} \bar{\zeta} + b_7 \bar{\eta} \bar{\zeta} + b_8 \bar{\xi} \bar{\eta} \bar{\zeta}$$

$$Z = c_1 + c_2 \bar{\zeta} + c_3 \bar{\eta} + c_4 \bar{\zeta} + c_5 \bar{\xi} \bar{\eta} + c_6 \bar{\xi} \bar{\zeta} + c_7 \bar{\eta} \bar{\zeta} + c_8 \bar{\xi} \bar{\eta} \bar{\zeta} \tag{B-6}$$

where the constants, a_j, b_j, c_j, $j = 1, \ldots, 8$ are determined by the corresponding values of the coordinates at the vertices in physical space according to Eq. (B-4). Equation (B-6) is valid for any point in the interior of the hexahedron. Thus, since the (X, Y, Z) coordinates of P are known, we have a system of equations for the coordinates of P in interpolation space. The above mapping must be one-to-one (i.e., the inverse mapping must exist). The mathematical requirement is that the warped hexahedron be "convex" (i.e. not too warped). For our applications, this requirement should be implicitly satisfied since the transformation to computational space maps the warped hexahedrons to cubes and is one-to-one.

Solution for the $(\bar{\xi}, \bar{\eta}, \bar{\zeta})$ corresponding to P is accomplished iteratively by applying Newton's method. Let the system be written as

$$\vec{X} = \vec{G}(\bar{\xi}, \bar{\eta}, \bar{\zeta}) = \vec{G}(\vec{\bar{\xi}})$$

Let

$$F(\vec{X}, \vec{\bar{\xi}}) = \vec{G}(\vec{\bar{\xi}}) - \vec{X} = 0$$

Newton's method gives

$$\vec{\bar{\xi}}^{\,\nu+1} = \vec{\bar{\xi}}^{\,\nu} - [\vec{F}_{\vec{\bar{\xi}}}(\vec{X}, \vec{\bar{\xi}}^{\,\nu})]^{-1} \bullet \vec{F}(\vec{X} \vec{\bar{\xi}}^{\,\nu}) \tag{B-7}$$

for the iteration, where

$$\vec{F}_{\vec{\bar{\xi}}} = \frac{\partial F_i}{\partial \bar{\xi}_j} = M_{ij}$$

and

$$M = \begin{bmatrix} (a_2 + a_5\bar{\eta} + a_6\bar{\zeta} + a_8\bar{\eta}\bar{\zeta}) & (a_3 + a_5\bar{\xi} + a_7\bar{\zeta} + a_8\bar{\xi}\bar{\zeta}) & (a_4 + a_6\bar{\xi} + a_7\bar{\eta} + a_8\bar{\zeta}\bar{\eta}) \\ (b_2 + b_5\bar{\eta} + b_6\bar{\zeta} + b_8\bar{\eta}\bar{\zeta}) & (b_3 + b_5\bar{\xi} + b_7\bar{\zeta} + b_8\bar{\xi}\bar{\zeta}) & (b_4 + b_6\bar{\xi} + b_7\bar{\eta} + b_8\bar{\zeta}\bar{\eta}) \\ (c_2 + c_5\bar{\eta} + c_6\bar{\zeta} + c_8\bar{\eta}\bar{\zeta}) & (c_3 + c_5\bar{\xi} + c_7\bar{\zeta} + c_8\bar{\xi}\bar{\zeta}) & (c_4 + c_6\bar{\xi} + c_7\bar{\eta} + c_8\bar{\zeta}\bar{\eta}) \end{bmatrix} \tag{B-8}$$

M is the Jacobian of the isoparametric transformation. Hence, M^{-1} must exist as long as the mapping is one-to-one. Since M is 3 by 3, its inverse can be computed directly as

$$M^{-1} = \begin{bmatrix} (M_{22}M_{33} - M_{23}M_{32}) & -(M_{12}M_{33} - M_{13}M_{32}) & (M_{12}M_{23} - M_{13}M_{22}) \\ -(M_{21}M_{33} - M_{23}M_{31}) & (M_{11}M_{33} - M_{13}M_{31}) & -(M_{11}M_{23} - M_{13}M_{21}) \\ (M_{21}M_{32} - M_{22}M_{31}) & -(M_{11}M_{32} - M_{12}M_{31}) & (M_{11}M_{22} - M_{12}M_{21}) \end{bmatrix} \frac{1}{\det M} \tag{B-9}$$

where

$$\begin{aligned} \det M = & -(M_{11}M_{22}M_{33} + M_{12}M_{23}M_{31} + M_{13}M_{21}M_{32}) \\ & + (M_{13}M_{22}M_{31} + M_{12}M_{21}M_{33} + M_{11}M_{23},M_{32}) \end{aligned} \tag{B-10}$$

The function $\vec{F}(\vec{x}, \bar{\xi})$ is

$$\begin{aligned} F_1 &= (a_1 - x) + a_2\bar{\xi} + a_3\bar{\eta} + a_4\bar{\zeta} + a_5\bar{\xi}\bar{\eta} + a_6\bar{\xi}\bar{\zeta} + a_7\bar{\eta}\bar{\zeta} + a_8\bar{\xi}\bar{\eta}\bar{\zeta} \\ F_2 &= (b_1 - y) + b_2\bar{\xi} + b_3\bar{\eta} + b_4\bar{\zeta} + b_5\bar{\xi}\bar{\eta} + b_6\bar{\xi}\bar{\zeta} + b_7\bar{\eta}\bar{\zeta} + b_8\bar{\xi}\bar{\eta}\bar{\zeta} \\ F_3 &= (c_1 - z) + c_2\bar{\xi} + c_3\bar{\eta} + c_4\bar{\zeta} + c_5\bar{\xi}\bar{\eta} + c_6\bar{\xi}\bar{\zeta} + c_7\bar{\eta}\bar{\zeta} + c_8\bar{\xi}\bar{\eta}\bar{\zeta} \end{aligned} \tag{B-11}$$

Typically, $\bar{\xi}^0 = (1/2, 1/2, 1/2)$ and the iteration converges to an rms residual of 10^{-4} in about five steps. The values of $(\bar{\xi}, \bar{\eta}, \bar{\zeta})$ are stored in arrays DXI, DYI, and DZI in PEGSUS.

They are reordered for use in XMER3D where they are called DXINT, DYINT, and DZINT.

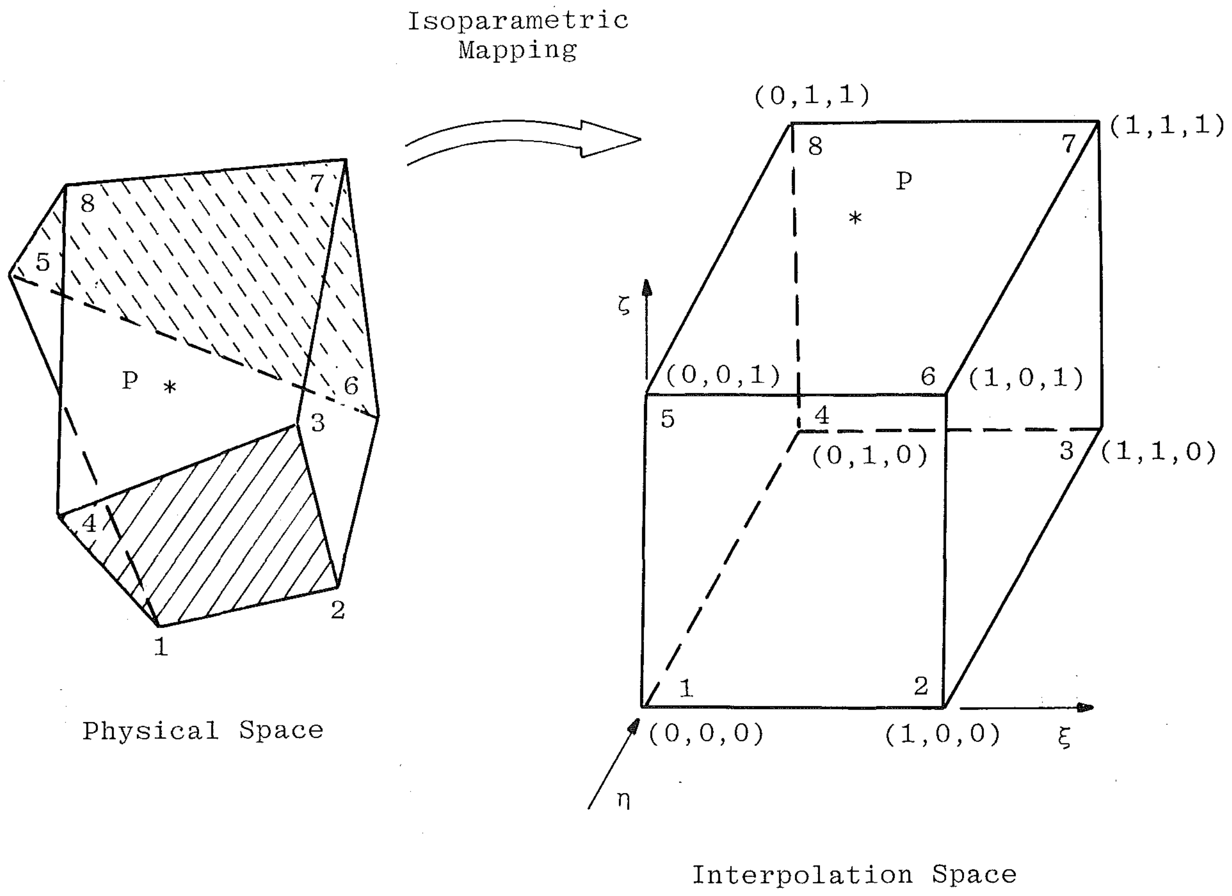


Figure B-1. Isoparametric mapping used for trilinear interpolation.

APPENDIX C
DATA STRUCTURES

PEGSUS

The Embedding Hierarchy

The embedding hierarchy establishes how the component grids are allowed to interact. It also determines the form of the data structure. The experience obtained from the 2-D chimera work (Refs. 15 and 21) shows that a less restrictive hierarchy will be required. In particular, an embedded grid must be allowed to overlay solid boundaries in the grid in which it is embedded. This extension means that holes may be introduced into a grid, $G_{l,i}$, not only from the embedded mesh, $G_{l+1,j}$, but also from the mesh in which it is embedded, $G_{l-1,k}$. Additionally, grids on the same level of the hierarchy must be allowed to interact or become linked. These requirements were kept in mind when the data structures were designed for the 3-D implementation. Therefore, the data structures are described for the more general case but are illustrated for the restricted case.

The restricted hierarchy described in Section 4.1 is illustrated in Fig. C-1. The notation $G_{l,i}$ is used to indicate the i^{th} grid on level l. We now introduce additional nomenclature that will be related to the data structure. The mesh $G_{l,i}$ is a precursor to its descendent grid G_{l+1},j, which is embedded within it. For example, in Fig. C-1 mesh G_{22} had G_{31} and G_{33} as descendants and G_{11} as its precursor. To account for these relationships, the arrays PRECUR and DECEND are introduced. To allow for a general structure of relationships among the grids, they are stored in an arbitrary order in memory and are assigned a mesh number. Only the root grid, G_{11}, has a predetermined number and it is one (1). The embedding hierarchy of Fig. C-1 with assigned mesh numbers is shown in Fig. C-2. The introduction of the pointers also simplifies the construction of lists and pointers.

Hierarchy Pointers

Because each grid has a unique number, M, assigned to it, any mesh is identified by a single number. For example, from Fig. C-2,

$$M = 3 = G_{22}$$

A grid's relationship to other grids in the hierarchy can be determined by specifying grids embedded within it (descendants) and the grid in which it is embedded (precursor). We define arrays to store precursor and descendent mesh numbers. These are PRECUR(M) which contains the mesh number of grid M, NDCEND(M) which stores the number of descendants of M, and DECEND(M,N) which holds the mesh number of the Nth descendant of M. The PRECUR array points to grids which are in lower levels of the hierarchy and DECEND points to grids which are in higher levels of the hierarchy.

The cartesian coordinates are stored in single arrays X(I), Y(I), and Z(I) to minimize storage. A pointer, IXPNTR(M), is used to store the value of the index I corresponding to the first element of mesh M. The maximum values of the indices (j,k,l) are stored in the arrays MJMAX(M), MKMAX(M), and MLMAX(M). If the address of the starting elements of the arrays X, Y, and Z are passed to subroutines via argument lists, the coordinates of any point in M are

$$\begin{matrix} X(J,K,L) \\ Y(J,K,L) \\ Z(J,K,L) \end{matrix} \left\{ \begin{matrix} J \in [1, MJMAX(M)] \\ K \in [1, MKMAX(M)] \\ L \in [1, MLMAX(M)] \end{matrix} \right.$$

The arrays X(I), Y(I), and Z(I) are ordered lists.

Thus, the pointer IXPNTR becomes

$$\begin{aligned} M = 2,\ & IXPNTR(1) = 1 \\ 2,\ & IXPNTR(2) = MJMAX(1)*MKMAX(1)*MLMAX(1) + IXPNTR(1) \\ 3,\ & IXPNTR(3) = MJMAX(2)*MKMAX(2)*MLMAX(2) + IXPNTR(2) \end{aligned}$$

Extension of the Hierarchy and Data Structure

Even limited experience with the chimera scheme has shown the method to have a significant potential to simplify grid generation. This potential can be increased by an extension of the hierarchy to allow grids on the same level to intersect. Relaxation of the hole generation restriction to allow an embedded grid, $G_{l+1,j}$ to have a hole introduced by a solid boundary in the precursor grid, $G_{l,i}$ requires only a slight change in the data structure and composite grid construction algorithm.

The increase in efficiency gained by allowing grids on the same level to intersect may be illustrated by the following example. The simple hierarchy shown in Fig. C-3 leads to a composite mesh such as that illustrated. The restriction to disjoint grids on the same level requires the wing grid, G_{31}, to be embedded in the fuselage grid. The total number of points could be reduced by relaxing this restriction (Fig. C-4). The complications that can be expected from the extension of the hierarchy are illustrated in Fig. C-4. They entail a hole crossing both grid boundaries and levels of the hierarchy.

The modification to the data structure to accommodate overlapping grids is the addition of a pointer to grids on the same level of the hierarchy which intersect a given grid. We introduce the pointer LINK(M,N) to store mesh numbers of the N^{th} grids intersecting or linking mesh M. The total number of such grids for mesh M is stored in the array NLINK(M). A similar structure is introduced to account for the holes—the pointer HOLES(M,N) to store mesh numbers of the N^{th} grid which introduces a hole into M, and the array NHOLES(M) to record the total number of grids which cause holes in M. The modifications will provide the capability to allow very general interactions among the grids.

Once the data structure is modified, the algorithm for constructing the composite mesh must be altered. The requirement is that additional searches be made of more grids to locate appropriate interpolational stencils. The above modifications are underway.

Boundary and Interpolation Data

The form of the data structure used for the boundary interpolation data depends upon how the data are collected. The procedure obtains hole data for all the grids and then generates outer-boundary data for all the grids. The data structure must associate the interpolated boundary point in a mesh M with the corresponding stencil in mesh M1. It must also associate the interpolation stencil in M with the corresponding boundary point in mesh M2.

The indices of the interpolated boundary points and the corresponding interpolation stencil reference point (See Appendix B) are stored in separate lists for each mesh. For simplicity, double-dimensioned arrays are used, JBPT(M,I), KBPT(M,I), and LBPT(M,I) for boundary points and JI(M,I), KI(M,I), and LI(M,I) for the stencil reference point. The arrays are filled as follows: Mesh M1 is searched for an interpolation stencil for a boundary point in grid M. When the stencil is located, the stencil reference point indices are stored in the lists JI(M1,I), KI(M1,I) and LI(M1,I) and the interpolation coefficients (See

Appendix B) are stored in the lists DXI(M1,I), DYI(M1,I), and DZI(M1,I). For convenience, the boundary point indices of the point in M are stored in the lists JBPT(M1,I), KBPT(M1,I), and LBPT(M1,I). The lists organize the data by the mesh number of the grid which contains the interpolation stencil. The total number of boundary points interpolated from mesh M is IBPTS(M). Thus, the lists JI, KI, LI, DXI, DYI, and DZI for mesh M contain information obtained from mesh M, whereas the data in the lists JBPT, KBPT, and LBPT for M are indices of points which belong to other grids.

The collection and data storage procedure automatically associate an interpolation stencil to the proper boundary point by the mesh number of the stencil. However, additional pointers are needed to sort the data according to the mesh number of the boundary point. Because PEGSUS first collects the data for all the hole boundaries and then all the outer boundaries, the lists for each mesh are naturally divided into sublists which correspond to separate boundaries of other grids (Fig. C-5). An additional set of pointers identifies the sublists; IPNTR(M,N) points to the first member, and NPNTR(M,N) points to the last member of the Nth boundary interpolated from mesh M. The total number of sublists or subsets for M is NSETS(M). Figure C-5 illustrates how the pointers are related to the interpolation data lists.

The bookkeeping is completed by providing a means of isolating a particular hole boundary or outer boundary. Consider the system of embedded grids given in Figs. C-1 and C-2. The grids are embedded according to the hierarchy allowed by PEGSUS. Each embedded grid is disjoint with respect to other grids on the same level of the hierarchy and contained completely within a single mesh on the next lower level of the hierarchy. Suppose we wish to examine the hole boundary in G_{22} (M = 3) caused by G_{32} (M = 5), according to the adopted storage convention, the indices of the hole boundary are contained in a subset or sublists of the points interpolated from mesh M = 5 [That is, G_{32} is the mesh from which values will be interpolated for points in G_{22} (M = 3) on the hole boundary caused by G_{32}]. Thus, all that is required is to locate the particular subset, say N, and the data will be contined in the lists between

$$\mathrm{IPNTR}(5,N) \leq I \leq \mathrm{NPNTR}(5,N)$$

We introduce a new pointer MHB(M,M1) to serve as a cross index for the subsets of the mesh interpolation lists. Suppose we wish to locate the subset number, ISET, of the hole-boundary data of points in M caused by the embedded grid, M1, then

$$\mathrm{ISET} = \mathrm{MHB}(M, M1)$$

If M1 does not introduce a hole into M, then ISET = 0. For the restricted hierarchy of Fig. C-6, only the descendants of mesh M need to be searched. Thus,

$$M1 = DECEND(M,N), \qquad N = 1...,NDCEND(M)$$

Figure C-6 illustrates the structure of MBH for the hierarchy of Fig. C-2.

In the example, the required subset is

$$ISET = MHB(M,M1) = N$$

where M = 3, M1 = 5. The desired boundary indices are located in the lists JBPT(M1,I), KBPT(M1,I), LBPT(M1,I) between the indices

$$IPNTR(M1,N) \leq I \leq NPNTR(M1,N)$$

A similar procedure is used to locate outer-boundary data for mesh M in the interpolation lists of the precursor mesh M1. The appropriate sublist in M1 is

$$ISET = MOB(M,M1)$$

where MOB is the outer-boundary cross-index pointer. Note that no alterations are required to MHB and MOB for the extensions described in the section of this appendix entitled "Extension of the Hierarchy and Data Structure."

XMER3D

The code XMER3D contains the flow solver or solvers. It serves the executive functions of controlling input, output, directing the solution on each mesh to the appropriate flow solver, and regulating the number of iterations performed on a mesh before proceeding to the next. However, there are only two functions that XMER3D must perform on the interpolation data. The first is to update interpolation boundaries of a mesh; the second is to interpolate data for the boundaries of embedded grids. Therefore, PEGSUS reorganizes the interpolation data for each grid into two sets of lists for use in XMER3D.

The first set contains the indices of the interpolation stencil reference points. JI(I), KI(I), and LI(I) and corresponding interpolation coefficients, DXI(I), DYI(I), and DZI(I). (Note the change in notation.) There are IIPNTS points which require interpolation from mesh M. The second set holds the list of indices of points in M that have values interpolated from other grids, JB(I), KB(I), and LB(I). Because all the interpolated values are retained in memory in a single list, QBC, a cross-index list, IBC(I), is also included in the second set of lists. There are IBPNTS points in the second set of lists for each mesh and IITOT points in QBC. Figure C-7 illustrates the structure of the lists for mesh M.

The data management in XMER3D maintains the grid, interpolation lists, and update lists on separate external units. XMER3D reads the appropriate data into memory as required. The management strategy minimizes the storage required for solution at the expense of more I/O overhead. In order to reduce the complexity of the data management, all the interpolated values in the list QBC permanently reside in memory. To minimize the storage requirement, the interpolated values are stored contiguously (Fig. C-8). For each grid a pointer, IISPTR, points to the element of QBC which corresponds to the first element in the list for mesh M. Storage in the list is arranged such that once the solution is advanced on mesh M and the required interpolations performed, the new values are stored by grid in QBC (i.e., one subset or sublist for each mesh). The storage strategy requires that a mechanism be provided which will allow the QBC list to be sorted to locate the proper values to update the interpolated boundaries of M. The required sorting information is supplied by the list IBC. Its function is to provide the index, I, in QBC which corresponds to a given boundary point (JB,KB,LB) in M. Thus, the data required to update (JB,KB,LB) in M are stored in QBC(I) (See Fig. C-8).

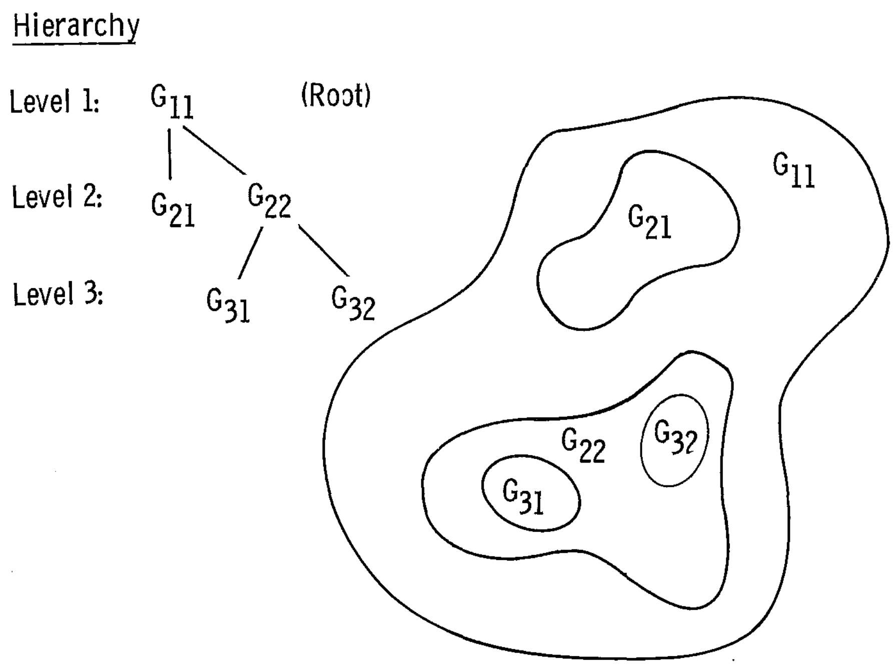


Figure C-1. Embedding hierarchy and graph.

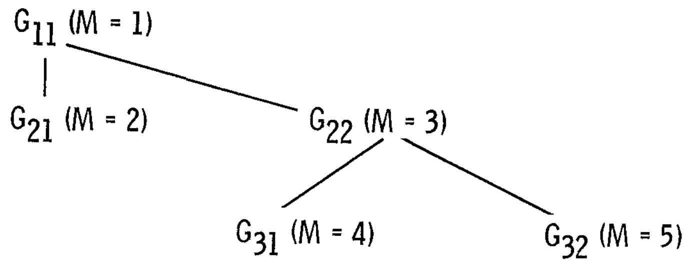


Figure C-2. Embedding hierarchy and associated mesh numbers.

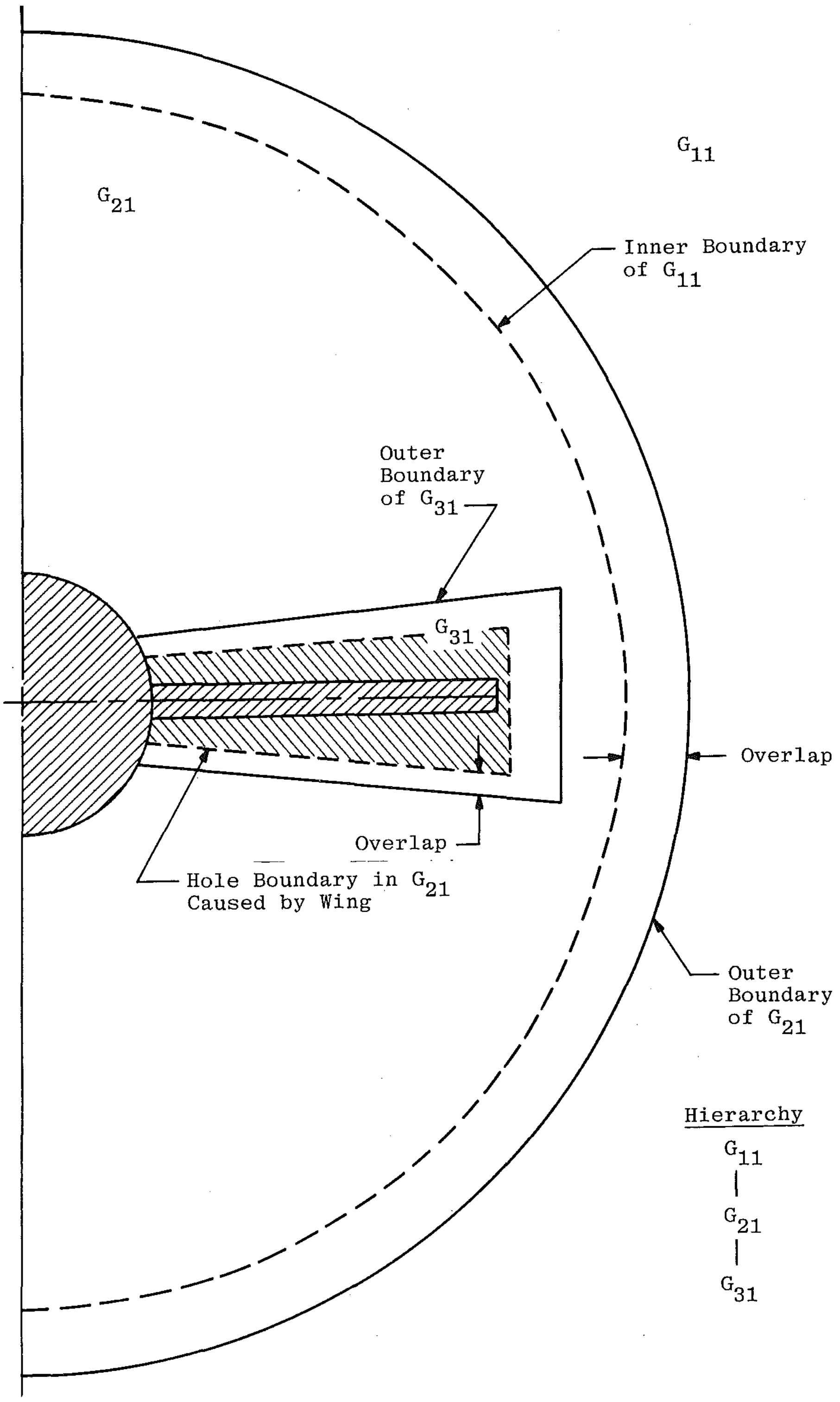

G_{11}
G_{21}
Inner Boundary of G_{11}
Outer Boundary of G_{31}
G_{31}
Overlap
Overlap
Hole Boundary in G_{21} Caused by Wing
Outer Boundary of G_{21}
Hierarchy
G_{11}
G_{21}
G_{31}

Figure C-3. Example of composite mesh for restricted embedding hierarchy.

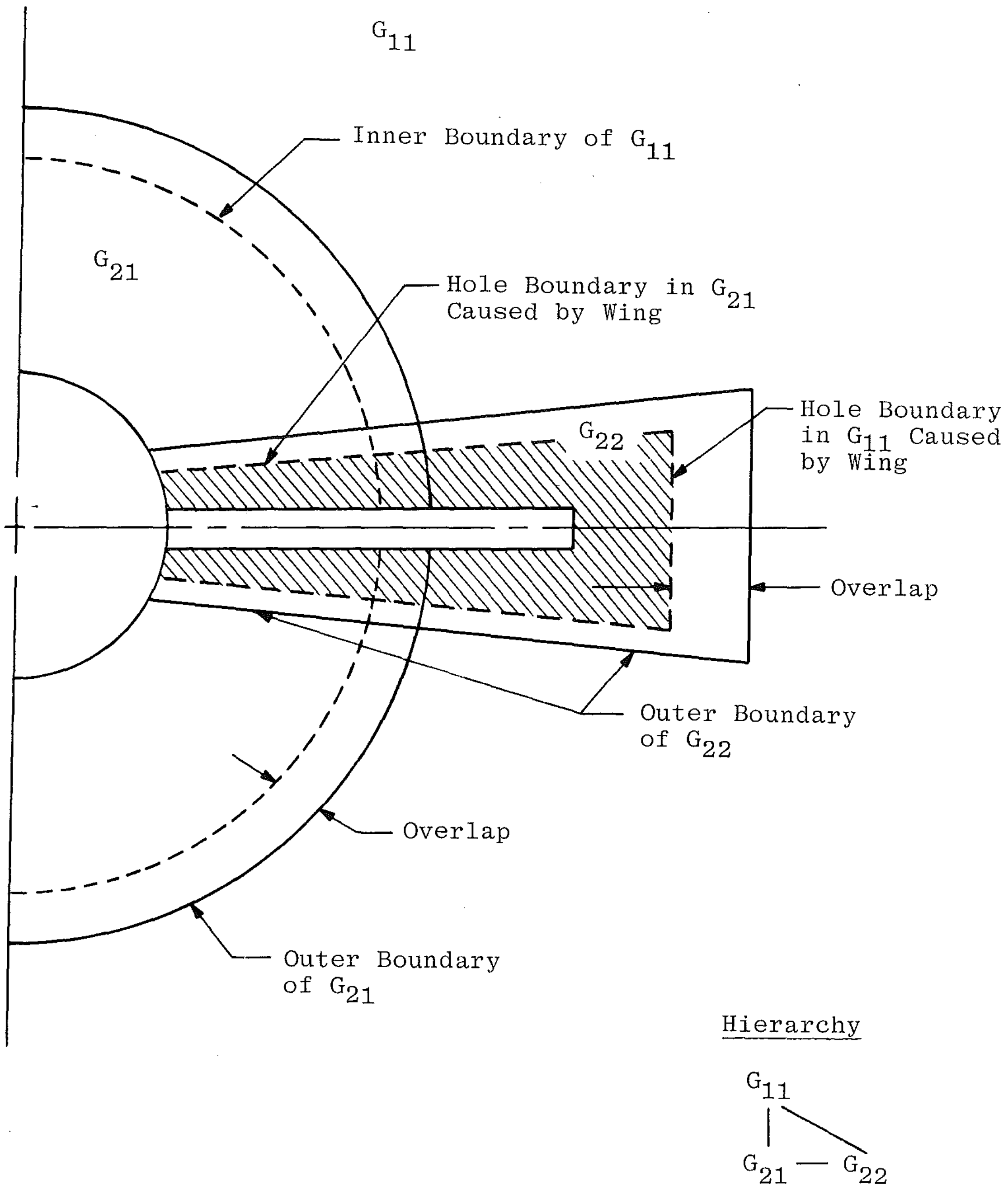

G_{11}
Inner Boundary of G_{11}
G_{21}
Hole Boundary in G_{21} Caused by Wing
Hole Boundary in G_{11} Caused by Wing
G_{22}
Overlap
Outer Boundary of G_{22}
Overlap
Outer Boundary of G_{21}
Hierarchy
G_{11}
G_{21} — G_{22}

Figure C-4. Example of a composite mesh for embedding hierarchy allowing intersecting grids.

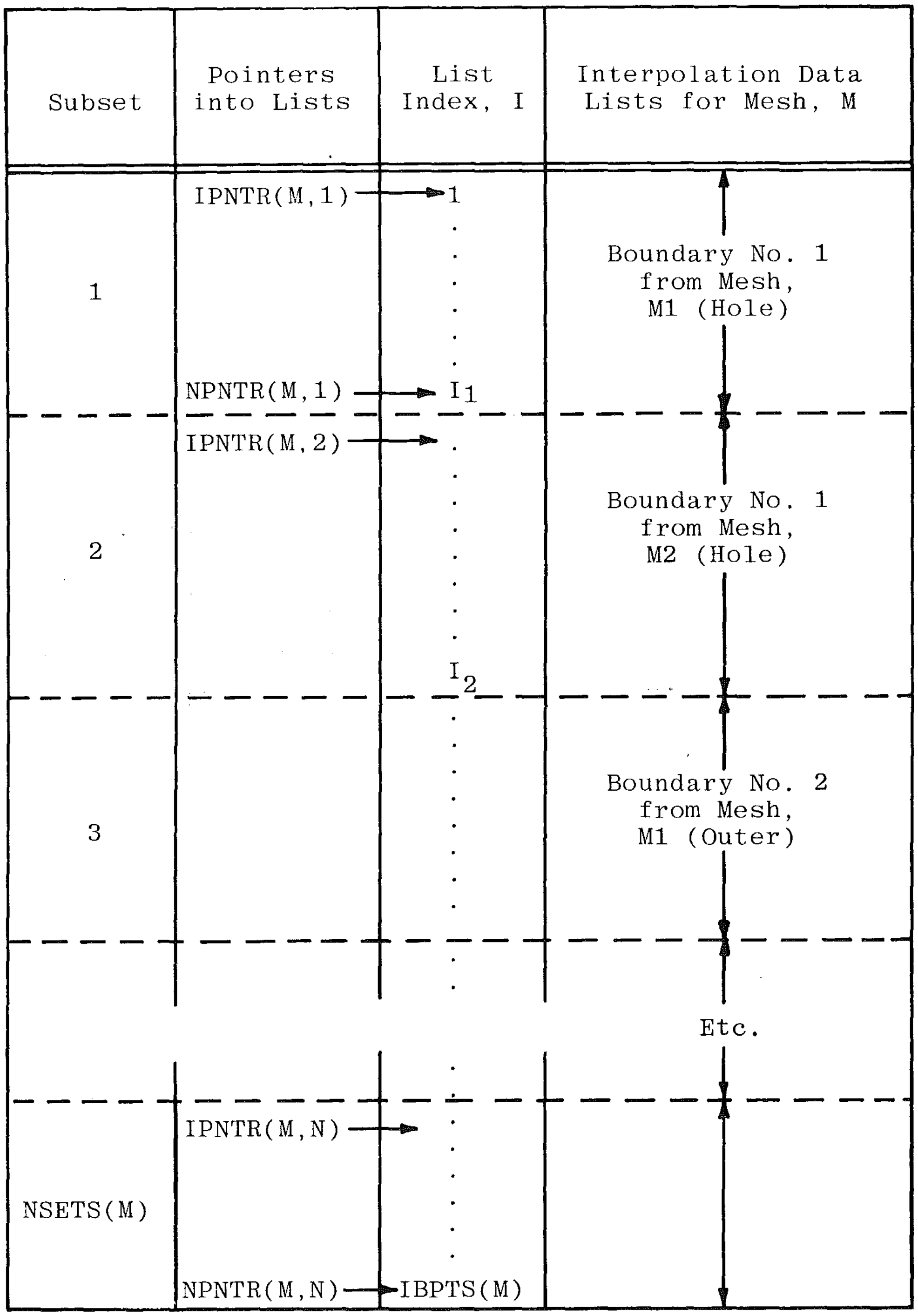

Subset
Pointers into Lists
List Index, I
Interpolation Data Lists for Mesh, M
1
IPNTR(M,1)
1
Boundary No. 1 from Mesh, M1 (Hole)
NPNTR(M,1)
I1
2
IPNTR(M,2)
Boundary No. 1 from Mesh, M2 (Hole)
I2
3
Boundary No. 2 from Mesh, M1 (Outer)
Etc.
NSETS(M)
IPNTR(M,N)
NPNTR(M,N)
IBPTS(M)

Figure C-5. Pointers into interpolation data lists used in PEGSUS.

MHB (M,M1) =

$M \backslash M_1$	1	2	3	4	5
1	0	X	X	0	0
2	0	0	0	0	0
3	0	0	0	X	N
4	0	0	0	0	0
5	0	0	0	0	0

Figure C-6. Matrix structure of cross-index array, MHB, for hole boundaries.

Interpolation List

Index	JI KI LI	DXI DYI DZI
1		
⋮		
IIPNTS		

Update List

Index	JB KB LB	IBC
1		
⋮		
IBPTS		

Figure C-7. Structure of interpolation data lists used in XMER3D.

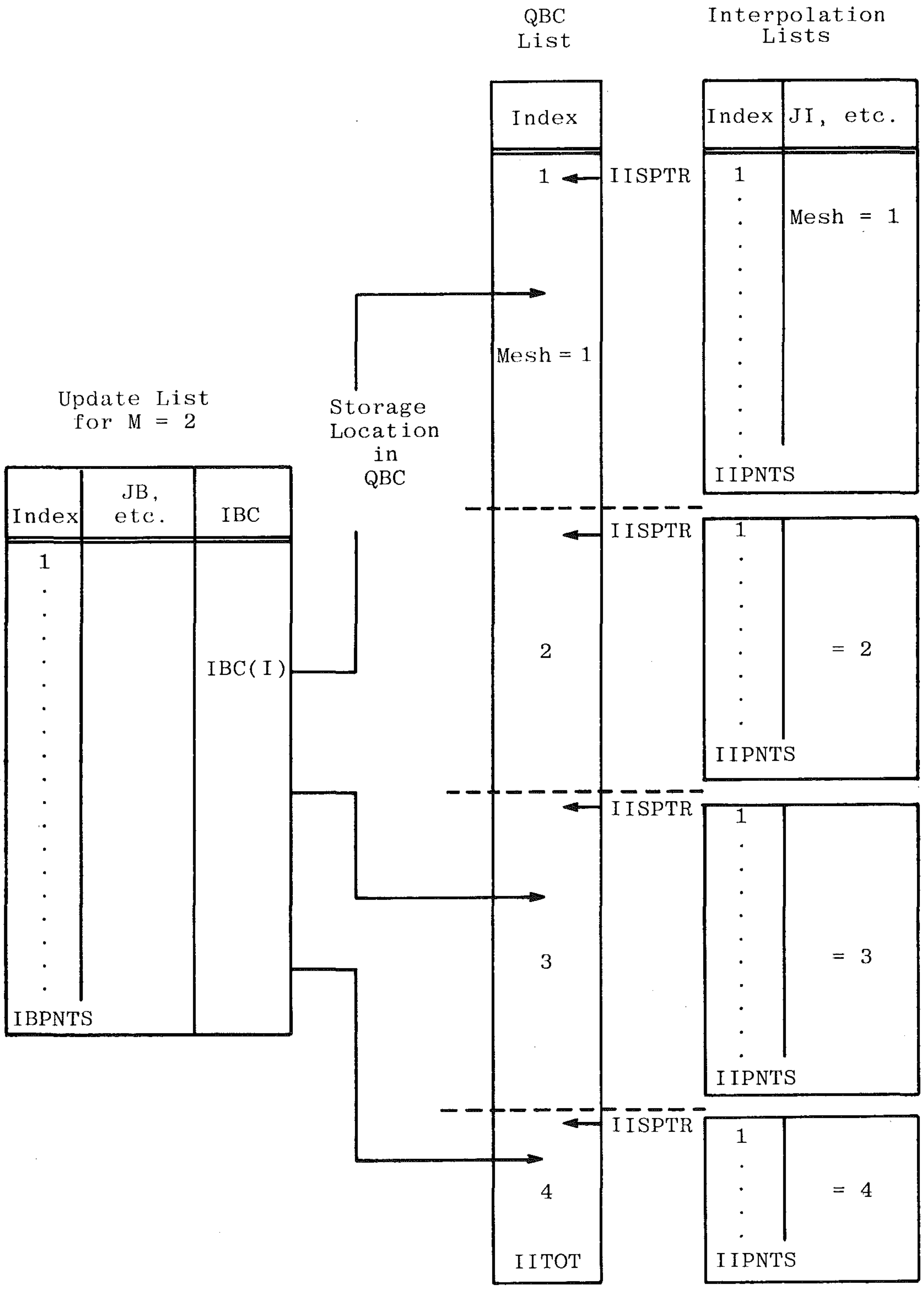


Figure C-8. Summary of data structure used in XMER3D.

APPENDIX D SUBROUTINE DESCRIPTIONS

Number†	Subroutine Descriptions
CHKOUT S10	Checks the interpolation stencils to locate those which do not contain the interpolated point. Trilinear interpolation requires the interpolation coefficients to take values in the interval [0,1]. If any value is outside the interval by more than ϵ (= 0.0001) the point is flagged.
CHKPLT S5	Plots specified surfaces of the input grids as a check. See namelist CKPLOT.
CHKSTN S11	Checks points in the interpolation stencil to determine if they contain interpolated data. For each point in the lists JI(M,I), KI(M,I), LI(M,I), and associated stencils of mesh M, sublists of JBPT(M1,I), KBPT(M1,I), and LBPT(M1,I) corresponding to points in M are searched to locate common indices.
CINDEX S12	Constructs the cross-index array IBC and the update list of boundary points, then computes the total number of points IBPNTS and IIPNTS in the update and interpolation lists for each mesh.
COMPOS S2	Supervises the construction of the composite grid from the component grids. The hierarchy specifications and component grids are input; the component grids transformed; composite grid points set; and the composite grid written to external storage for input to XMER3D.
FRNGE S19	Constructs the fringe or boundary about the hole introduced by grid M1 (descendant in present hierarchy). The fringe points are identified by setting IBLANK = M1.
HDATA S14	Reads the namelist HBOUN which contains the specifications for the initial hole boundary for all grids. The assumption is that each descendent mesh causes only one hole.
HINTPT S16	Locates interpolation stencils in descendent mesh M1 for hole-boundary points in M. Trilinear interpolation is assumed.

† NOTE: Numbers correspond to subroutine numbers in Table A-1 in Appendix A.

Number	Subroutine Descriptions
HLOCAT S21	Identifies the points of mesh M interior to the initial hole boundary introduced from descendent mesh M1. Interior points are located by forming dot product of $\vec{R}_p$ and $\vec{N}$ where $\vec{R}_p$ is the position vector from the nearest point on the boundary to a field point of mesh M, and $\vec{N}$ is the corresponding surface outward unit normal. If the dot product is positive, the point is outside the hole. The search is restricted to points within a sphere whose origin is the mean value of the coordinates of the initial surface and radius equal to the maximum from the sphere origin to the farthest surface point (See Section 3.2).
HOLE S6	Supervises the construction of holes and computation of the associated interpolation data for all grids. The construction procedure sets IBLANK = 0 at interior points and boundary points.
INITHB S18	Constructs the initial hole boundary. The boundary coordinates are stored in 2-D arrays.
INITIA S1	The initial values of the code parameters are set (also BLOCKDATA), and the title, hierarchy data in namelist HIERCY, and search limits in namelist SEARCH are read. Summaries of the input values are written to unit 6.
INTDAT S15	Computes the interpolation coefficients for trilinear interpolation using Newton's method.
MAXMIN	Determines the maximum and minimum values of component grid coordinates for plotting purposes.
NEARPT S24	Locates the nearest point in mesh M to a specified point.
NEWTON	Solves the trilinear interpolation equations for the coordinates of the interpolated point in interpolation space (i.e. $\bar{\xi}$, $\bar{\eta}$, $\bar{\zeta}$). For details, see Appendix B.
NORMAL S25	Computes the outward unit normal to a specified surface by constructing the surface tangents $\vec{T1}$ and $\vec{T2}$ and forming the cross-product $\vec{T1} \times \vec{T2}$.
NUMHOL	Counts the total number of points within holes (including fringe points) in the composite grid.

Number	Subroutine Descriptions
OBOUN S28	Loads the outer-grid boundary into 2-D arrays.
ODATA S26	Reads the namelist OBOUND which contains the specifications for all the component grid outer boundaries.
OLOCAT S27	Locates interpolation points for the outer boundary of mesh M by searching the precursor grid M1 for the nearest point corresponding to each boundary point.
OUTER S7	Supervises the computation of interpolation data for the outer boundaries of embedded grids. The outer-boundary specifications are input; the interpolation data computed; and points are set.
OUTPUT S3	Supervises the final check on and output of interpolation data. It also writes all the final summaries for the composite grid and makes estimates of storage parameters used in XMER3D.
PLANE	Plots a constant surface of J, K, or L depending upon the value of the flag ICASE, which is set in the calling routine.
PLTHI	Supervises the plotting of hole boundary and corresponding interpolation stencil reference points.
PLTOI S29	Supervises the plotting of outer-boundary and corresponding interpolation stencil reference points.
PLTHOL S17	Plots the initial hole boundary in mesh M caused by the descendent mesh M1.
PLTIBL S20	Plots the final hole boundary in mesh M caused by all its descendants. The plot is made in computational space.
PLTINT	Plots the hole boundary and interpolation point by connecting them with a line segment.

Number	Subroutine Descriptions
QUAD S23	Designates the interpolation reference point by identifying the point (JMIN, KMIN, LMIN) of the cube containing the interpolated point. The reference point is selected based on a transform to the uniform computational space. Note that INTDAT performs additional checks to ensure that the cube specified by the reference point actually contains the interpolated point (See Appendix B).
RGRID S8	Reads a grid from the external unit MESH + 10 and checks the consistency of the data input from namelist GRDPRM with similar data on the external unit.
SETPTR S22	Sets the grid pointers MHB, MOB, IPNTR, and NPNTR; loads the lists NSETS, IBPTS, JBPT, KBPT, LBPT, JI, KI, LI, DXINT, DYINT, and DZINT.
TRANS S9	Transforms an input component grid by translating, rotating, and scaling the coordinates. The rotations are assumed to be applied in the following order: z-axis (pitch), y-axis (yaw), and x-axis (roll). It is very important to remember that all transformations are with respect to the composite grid origin.
WCOORD S4	Writes the composite grid coordinates to unit 1 in the format that is expected by the flow solver, XMER3D. The records contain the x, y, z coordinates for each grid, one grid at a time.
WIBLNK S13	Writes IBLANK to unit 2 in form expected by the flow solver, XMER3D. Each record will contain the IBLANK array for a single grid.

APPENDIX E
GLOSSARY OF GLOBAL VARIABLES

Variable	Description

ALFA(3)
BETA(3)
GAMA(3)
	Transformation Parameters. They are rotation angles of new coordinate axis (composite) grid relative to input axis system.
	
	ALFA — rotation about x-axis (deg)
	BETA — rotation about y-axis (deg)
	GAMA — rotation about z-axis (deg)

DECEND(M,N)	**Hierarchy parameter.** It is an integer pointer which points to mesh number of the Nth descendant of mesh M.

DXINT(M,I)
DYINT(M,I)
DZINT(M,I)
	Interpolation Variables. They are lists which contain the interpolation coefficients for the trilinear interpolation of boundary points in mesh M (i.e. $\bar{\xi}$, $\bar{\eta}$, $\bar{\zeta}$).

IBC(I)	**XMER3D Bookkeeping.** IBC is a cross-index list that points to storage locations of interpolated values for boundary points (See Appendix D). It connects lists of boundary-point indices to the corresponding interpolated value.
IBMAX JBMAX	**Boundary Surface Variables.** These variables specify the maximum number of points in each surface coordinate direction (See VNX, VNY, VNZ, JB, etc., and JBO, etc.).
IBDIM JBDIM	**Code Parameters.** These parameters specify the maximum allowable values for IBMAX and JBMAX. They are array dimensions.
IBLANK(I)	**XMER3D Bookkeeping.** This is an array of flags for each grid point in each mesh. It takes the value of 1 for points exterior to the hole and 0 for points within or on the boundary of a hole. Note that points interior to a hole are excluded from the solution on that mesh. Hole points that are boundary points have values of the flow variables interpolated from other grids.

Variable	Description
IBPNTS	**XMER3D Bookkeeping.** This variable specifies the total number of boundary points in mesh that must be updated from values interpolated in other grids.
IBPTS(M)	**Interpolation Variable.** IBPTS contains the total number of boundary points interpolated on mesh M (See JBPTS, DXINT, JI, etc.).
ICKPLT	**Plot Parameter.** (Logical) If this plot flag value is TRUE, check plots of grid coordinate surfaces are to be made; if value is FALSE, no check plots are made (Also see JPLOTS, etc., and NPLOTS).
IDIM	**Code Parameter.** This parameter specifies the maximum allowable number of interpolation points for each mesh. It is used as an array dimension.
IFLAG(I)	**Work Array.** It is used with outer boundary surface index lists JBO, KBO, LBO to sort boundary interpolation points for linked grids.
IFORMT	**Input Format Parameter.** This parameter allows for multiple forms of the input format of component grids. Code currently has only one allowable format, hence IFORMT = 1 (See subroutine RGRID).
IHBTYP(M)	**Hole Boundary Specification Parameter.** This parameter specifies the topology and type of initial hole boundary to be specified. Permitted values are 110—warped spherical surface given by L = constant and J along lines of longitude; 120—warped hemisphere with base at J = JE; 210—warped cylindrical surface with L = constant surface and K along the cylinder axis. End planes included; 220—warped cylindrical surface with open end at K = KS; [Also see JH1(M), etc., and subroutine INITHB.]

Variable	Description
IIEPTR IISPTR	**XMER3D Bookkeeping.** These are pointers into lists of interpolated boundary data. They correspond to last and first element of list QBC for data interpolated in mesh M.
IIPNTS	**XMER3D Bookkeeping.** This variable specifies the number of boundary points interpolated from solution on mesh M.
IITOT	**XMER3D Bookkeeping.** It specifies the total number of points interpolated in the composite mesh.
IOBTYP(M)	**Outer-Boundary Specification Parameter.** This parameter specifies the topology and type of outer-boundary surface for mesh M. Permissible values are 110—warped spherical surface given by L = constant and J along lines of longitude; 120—warped hemisphere with base at J = J02; 130—warped hemispherical surface with open base at J = J02; 210—warped cylindrical surface with L = constant surface and K along the cylinder axis; end planes included; 220—warped cylindrical surface with open end at K = K01; [Also see J01(M), etc., and subroutine OBOUND.]
IPNTR(M,N)	**Interpolation Variables.** These are pointers into lists of interpolation stencil reference points, interpolation coefficients, and corresponding boundary-point lists. They specify the first and last index of points which belong to the Nth subset of the list. The points are members of grid M (See JBPT, etc.).
ITO	**Work Variable.** It is the number of points in the JNO, KNO, LNO, JBO, KBO, LBO arrays.
ITOTAL	**Work Variable.** It is the number of points in the JN, KN, LN arrays.

Variable	Description
ITRANS	**Transformation Parameter.** (Logical) This parameter specifies whether or not a transformation of input grid coordinates is required. If the value is TRUE, a transform is required; if it is FALSE, no transform is needed (See ALFA, BETA, GAMA, XO, YO, Z, SCALE).
IXPNTR(M)	**Bookeeping Parameter.** This is a pointer into the grid and IBLANK arrays. It points to the location of the first element in mesh M. Values of the flow variables at these points will be interpolated from M (Also see J1, K1, L1, IPNTR, NPNTR, NSETS).

JB(I)
KB(I)
LB(I)
	Work Arrays. They hold boundary-point indices of the current mesh.

JBO(I)
KBO(I)
LBO(I)
	Work Arrays. They contain boundary-point indices of outer boundaries. They are used with IFLAG(I) and are necessary to deal with linked grid outer boundaries.

JBPT(M,I)
KBPT(M,I)
LBPT(M,I)
	Interpolation Variables. They are lists of boundary-point indices that belong to boundaries of other grids which are embedded in mesh M. They specify the first and last index of the Nth subset of the list. The interpolation coefficients and corresponding boundary-point lists are DXINT, DYINT, DZINT, and JI, KI, LI.

JDIM KDIM LDIM	**Code Parameters.** They specify the maximum allowable values of JMAX, KMAX, and LMAX. They are used as array dimensions.

JH1(M)
KH1(M)
LH1(M)
JH2(M)
KH2(M)
LH2(M)
	Hole Boundary Specification Parameters. These variables specify the beginning and ending values of grid coordinates which specify the initial hole boundary caused by mesh M. Specific values depend upon grid topology (See subroutine INITHB description, IHBTYP and input description, Appendix F).

JI(M,I)
KI(M,I)
LI(M,I)
	Interpolation Variable. They are lists of interpolation stencil reference indices. The points belong to mesh M (See JBPT, KBPT, LBPT, IPNTR, NPNTR, and NSETS).

Variable	Description

JLH1(M,N)
KLH1(M,N)
LLH1(M,N)
JLH2(M,N)
KLH2(M,N)
LLH2(M,N)
	Hole-Boundary Specification Parameters. These variables specify the beginning and ending values of grid coordinates which specify the initial hole boundary used by the Nth grid linked with mesh M. Values depend upon grid topology [See JH1, etc., LHBYPM(M,N), and Appendix F.]

JN(I)
KN(I)
LN(I)
	Work Arrays. They contain indices of the interpolation stencil reference point in the current grid, M (Also see JB, KB, LB).

JO1(M)
KO1(M)
LO1(M)
JO2(M)
KO2(M)
LO2(M)
	Outer-Boundary Specification Parameters. These variables specify the beginning and ending indices to be used in defining outer-grid boundaries. Values depend on grid topology (See description of subroutine OBOUND, IOBTYP, and Appendix F).

JPLOT(M,N)
KPLOT(M,N)
LPLOT(M,N)
	Plot Specification Parameters. These variables specify constant surfaces of J, K, or L to be plotted for the Nth plot of mesh M (See ICKPLT and description of subroutine CHKPLT).

JRS1(M)
KRS1(M)
LRS1(M)
JRS2(M)
KRS2(M)
LRS2(M)
	Grid search parameters. They specify limiting values of grid indices to be used when searching for interpolation points contained in mesh M. Defaults are maximum grid dimensions (See Appendix F).

LHBTYP(M,M1)	**Hole-Boundary Specification Parameter.** This parameter specifies the initial hole-boundary type (See IHBTYP) for holes introduced into mesh M by the linked Mesh M1 (See JLHB1, etc.).
MDIM	**Code Parameter.** This parameter specifies the maximum allowable number of component grids. It is used as an array dimension.

Variable	Description
MESHN	**Input Parameter.** It is the mesh number assigned a priori to a component mesh. It is part of the data on the external file containing the input grid. It serves as an internal check on input data.
MESHNO	**Input Parameter.** It is the mesh number of the component grid whose parameters are contained in namelist GRDPRM. It is used as an internal check to verify the proper correspondence with the input component grid file.
MHBS(M,M1)	**Bookkeeping Parameter.** This pointer points to the subset number of the hole-boundary points of mesh M caused by mesh M1. In the present hierarchy, M is a descendent mesh (Also see NSET, IPNTR, NPNTR).

MJMAX(M)
MKMAX(M)
MLMAX(M)
	Hierarchy Parameter. These parameters contain the number of points in the three coordinate directions (ξ, η, ζ) of each mesh M in the hierarchy (See JDIM, KDIM, LDIM).

MOBS(M,M1)	**Bookkeeping Parameter.** This is a pointer to the subset number of the outer-boundary points of mesh M which are interpolated from values in M1. (Also, see IPNTR, NPNTR, and NSETS).
MPLOTS	**Plot Parameter.** This counter records the total number of plots made during an execution of PEGSUS.
NDCEND(M)	**Hierarchy Parameter.** This parameter contains the number of descendent grids of mesh M (See DCEND).
NLINK(M)	**Hierarchy Parameter.** This parameter specifies the number of grids linked to mesh M.
NMESH	**Hierarchy Parameter.** It specifies the total number of component grids in the composite mesh (See MDIM).
NPLOTS(M)	**Plot Parameter.** This variable is the total number of check plots made for mesh M.

Variable	Description
NPNTS	**Hierarchy Parameter.** This parameter is the total number of points in the composite grid.
NSETS(M)	**Bookkeeping Variable.** This variable has a value equal to the total number of boundaries requiring interpolation that are embedded in mesh M.
PRECUR(M)	**Hierarchy Parameter.** It is a pointer to the mesh number of the precursor grid of mesh M. In the present hierarchy, each mesh can have only a single precursor.
SCALE	**Tranformation Parameter.** It is a multiplicative scaling factor for input component grids.

VNX(I,J)
VNY(I,J)
VNZ(I,J)
	Boundary Surface Variables. These are the outward unit normal vectors to the surface stored in XB, YB, ZB (Also see IBMAX and JBMAX).

XB(I,J)
YB(I,J)
ZB(I,J)
	Boundary Surface Variables. These are the coordinates of the boundary surface (See IBMAX, JBMAX, and XB, etc.).

XFAC YFAC ZFAC	**Plot Parameters.** They specify the plot viewpoints.

XI(I)
YI(I)
ZI(I)
	Interpolation Variables (Work Arrays). They contain the interpolation coefficient data for points interpolated from values in mesh M.

XO YO ZO	**Transformation Parameters.** These are the unscaled coordinates for a translation of component grid coordinates.

APPENDIX F
DESCRIPTION OF INPUT AND OUTPUT

INTRODUCTION

Input to PEGSUS takes the form of binary data (i.e. the component grid data) and namelist input. This appendix details the formats required of the binary data, the namelists, associated variables, and their default values.

BINARY FILE INPUT

The default format for the component grid files is (IFORMT = 1 on unit number IUNIT = MESHN + 10)

Record Number	**Variable**
1	MESHN
2	JMAX, KMAX, LMAX
3	(((X(J,K,L), J = 1, JMAX), K = 1, KMAX), L = 1, LMAX),
	(((Y(J,K,L), J = 1, JMAX), K = 1, KMAX), L = 1, LMAX),
	(((Z(J,K,L), J = 1, JMAX), K = 1, KMAX), L = 1, LMAX)

where MESHN is the mesh number assigned a priori. This number is arbitrary except for MESHN = 1 which must be the global mesh. JMAX, KMAX, and LMAX are the maximum values of the J, K, and L indices (i.e. the number of points in each coordinate direction). These data are input in subroutine RGRID.

INPUT

A schematic of the input data is given in Fig. F-1, and a detailed description is contained in the following subsections. The figure illustrates the order of input and the subroutine in which it is read.

TITLE

TITLE is read on a 10A8 format in subroutine INITIA. It is an 80-character description of the composite grid.

HIERCY

HIERCY is a namelist and is read in INITIA. It contains the following parameters:

DECEND (M,N)	Mesh number of the Nth descendant of mesh M TYPE: INTEGER, Dimensions: MDIM × MDIM Default = 0.0
LINK (M,N)	Mesh number of the grid linked to mesh M (not used), Dimensions: MDIM × MDIM
MJMAX (M) MKMAX (M) MLMAX (M)	Number of points in J, K, and L coordinate directions for mesh M Dimension: MDIM
NDCEND (M)	Number of descendants of mesh M Dimension: MDIM Default = 0
NLINK (M)	Number of grids linked to mesh M (not used) Dimension: MDIM Default = 0
NMESH	Number of component grids Default = 1
PRECUR (M)	Mesh number of precursor of mesh M Type: INTEGER, Dimension: MDIM Default = 0

GRDPRM

GRDPRM is a namelist which is read in RGRID. A separate GRDPRM namelist is required for each component mesh. It contains the grid parameter specifications

IFORMT	Integer flag denoting the format of the component mesh input file; default and only acceptable current value is 1; this parameter allows multiple formats for binary grid files. Default: 1
ITRANS	Specifies need to transform component mesh; acceptable transforms are translation, rotation, and scaling; NOTE: All transforms are referenced to the composite grid coordinates (See Appendix E) Type: LOGICAL Default: FALSE. (i.e. no transformation)

ALFA(3)
BETA(3)
GAMA(3)
	The rotation angles in degrees of each coordinate axis of input grid relative to composite grid; the angles are associated with the axis: ALFA/x-axis, BETA/y-axis, and GAMA/z-axis (See Appendix E)
	Default: 0.0
	Dimension: 3

XO YO ZO	The origin shift of input grid in the input (unscaled) coordinates (See Appendix E) Default: 0.0
SCALE	Multiplicative scale factor Default: 1.0
MESHNO	Mesh number corresponding to data in GRDPRM; this number must match MESHN on component grid file

CKPLOT

CKPLOT is a namelist which is read in subroutine CHKPLT. The namelist contains specifications for plotting coordinate surfaces of the component grids. The parameters are

ICKPLT	Flag specifying that checkplots are to be made Type: LOGICAL Default: .FALSE. - no plots
NPLOTS (M)	Number of plots to be made from mesh M Dimension: MDIM Default: 0

JPLOT(M,N)
KPLOT(M,N)
LPLOT(M,N)
	A nonzero value specifies the surface to be plotted in the Nth plot from mesh M
	Only one coordinate may be nonzero for each plot
	Dimensions: MDIM × PLTDIM
	Default: 0

XFAC YFAC ZFAC	Magnification factors for coordinates of the view point for the plots; large values provide less perspective Default: 1000.0

HBOUN

HBOUN is a namelist which is read in subroutine HDATA. It contains specifications for initial hole boundaries. They are

IHBTYP (M)	Flag specifying topology and type of initial hole boundary. Currently acceptable values are 100—Warped spherical surface given by L = constant and J along lines of longitude; 110—Warped hemispherical surface as above with base at JH2(M);

	210—Warped cylindrical surface given by L = constant and K along cylinder axis; both end surfaces are included;
	220—Warped cylindrical surface as above with open end at K = KH1(M) Dimension: MDIM Default: 0
JH1(M) KH1 (M) LH1 (M) JH2 (M) KH2 (M) LH2 (M)	Ranges of indices defining surface; those ending with 1 are initial value, and those ending with 2 are final value of index; their significance depends on IHBTYP (M); typical values are IHBTYP = 110 JH1(M) = 1 JH2(M) = JMAX 120 KH1(M) = 1 KH2(M) = KMAX LH1(M) = 1 LH2(M) = LO, LO < LMAX = 210 JH1(M) = 1 JH2(M) = JMAX 220 KH1(M) = K1 KH2(M) = K2 LH1(M) = 1 LH2(M) = L2 where K1, K2 ϵ [2,KMAX] and L2 < LMAX
	NOTE: 1. The parameters JH1(M), etc., specify the hole boundary caused by mesh M in its precursor grid M1; 2. The linked grid logic is not included; 3. All the current boundaries can be put into a spherical surface coordinate. Dimension: MDIM Default: 0

OBOUN

OBOUN is a namelist read in subroutine ODATA and contains the specifications to be used to define the outer boundary of mesh M for the purpose of locating suitable interpolation stencils in other grids. The parameters are

IOBTYP(M)	Flag specifying topology and type of outer boundary; currently acceptable values are 110—Warped spherical surface given by L = constant and J along lines of longitude; 120—Warped hemispherical surface as above with base at JO2(M); 130—Warped hemispherical surface with open base at JO2(M); 210—Warped cylindrical surface given by L = constant and K the cylinder axis; both end surfaces are included 220—Warped cylindrical surface as above with open end at K = KO1(M) Dimension: MDIM Default: 0
JO1(M) KO1 (M) LO1 (M) JO2 (M) KO2 (M) LO2 (M)	Ranges of indices defining surface; those ending with 1 are initial value and those ending with 2 are final value of index; their significance depends on IOBTYP (M). Typical values are IOBTYP = 110 JO1(M) = 1 JO2(M) = JMAX 120 KO1(M) = 1 KO2(M) = KMAX LO1(M) = 1 LO2(M) = LO where LO < LMAX

```
=              210JO1(M) = 1  JO2(M) = JMAX
               220 KO1(M) = K1  KO2(M) = K2
                   LO1(M) = 1  LO2(M) = L2
```

where

$K1, K2 \in [2, KMAX]$ and
$L2 < LMAX$

BINARY OUTPUT FILES

PEGSUS generates two output files for input to XMER3D. They are the composite mesh, and the interpolation and bookkeeping data. The data on these files are organized by grid to facilitate separation into individual working files. The formats for each are described in the following subsections.

Composite Mesh File

This file is written on unit 1. Each grid is written separately.

Record Number	Format
1	MESH, JMAX, KMAX, LMAX
2	(((X(J,K,L), J=1, JMAX), K=1, KMAX), L=1,LMAX),
	(((Y(J,K,L), J=1, JMAX), K=1, KMAX), L=1,LMAX),
	(((Z(J,K,L), J=1, JMAX), K=1, KMAX), L=1,LMAX).

MESH = 2, 3, ...

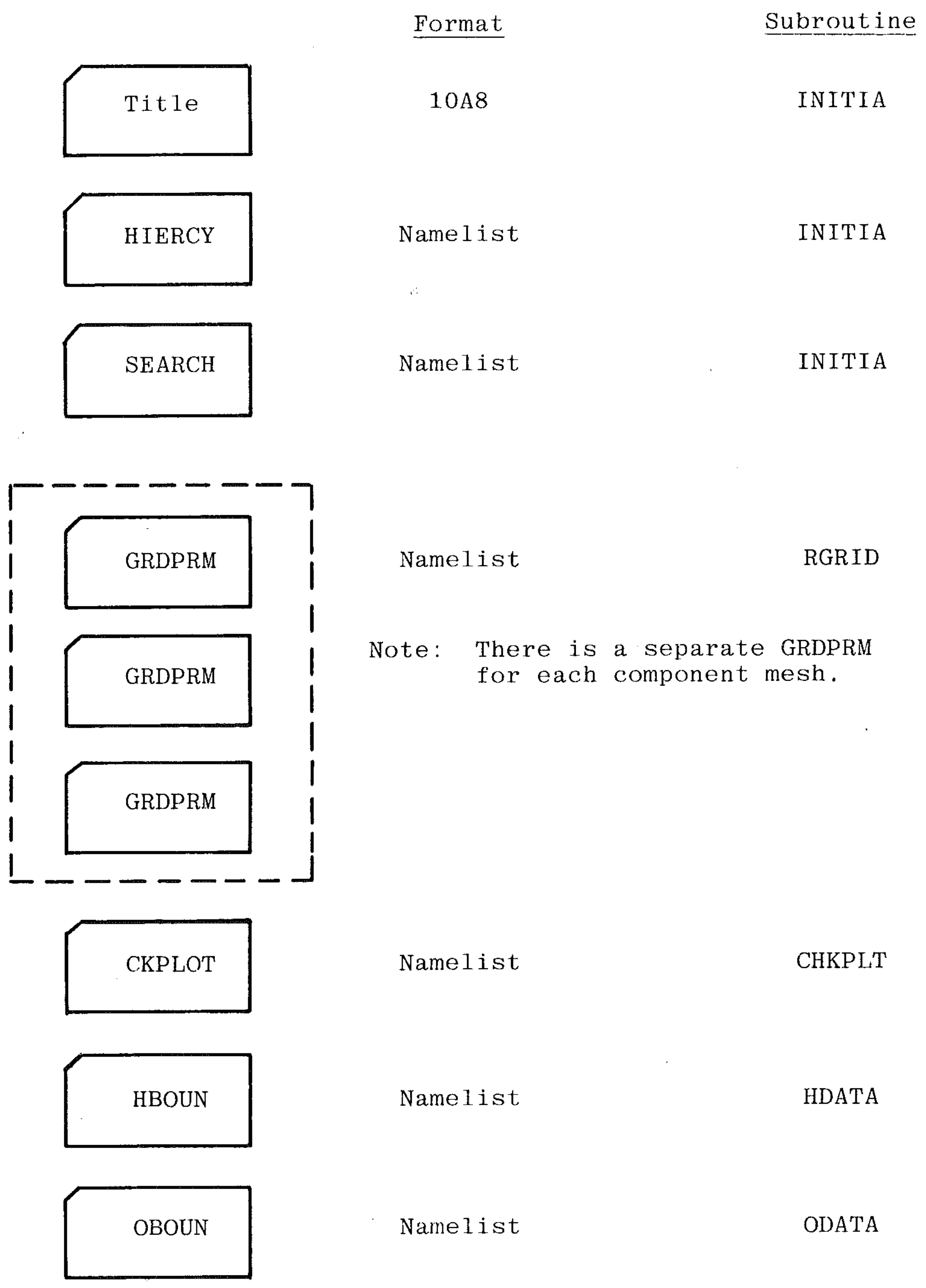
Interpolation and Bookkeeping Data File

This file is written to unit 2. The data for each grid are written separately.

Record Number	Format
1	IBPNTS, IIPNTS, IIEPTR, IISPTR

2	(JI(I), KI(I), LI(I), DXINT(I),DYINT(I)
	DIINT(I), I = 1, IIPNTS)
3	(JB(I), KB(I), LB(I), IBC(I), I=1, IBPNTS)
	(((IBLANK(J,K,L), J=1,JMAX),
	K=1,KMAX),L=1,LMAX)

MESH = 2, 3, ...



	Format	Subroutine
Title	10A8	INITIA
HIERCY	Namelist	INITIA
SEARCH	Namelist	INITIA
GRDPRM	Namelist	RGRID
GRDPRM		
GRDPRM		
CKPLOT	Namelist	CHKPLT
HBOUN	Namelist	HDATA
OBOUN	Namelist	ODATA

Note: There is a separate GRDPRM for each component mesh.

Figure F-1. Input data for PEGSUS.